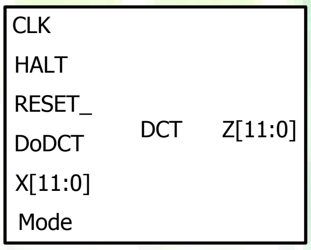
Lab 1 Case study: RTL design

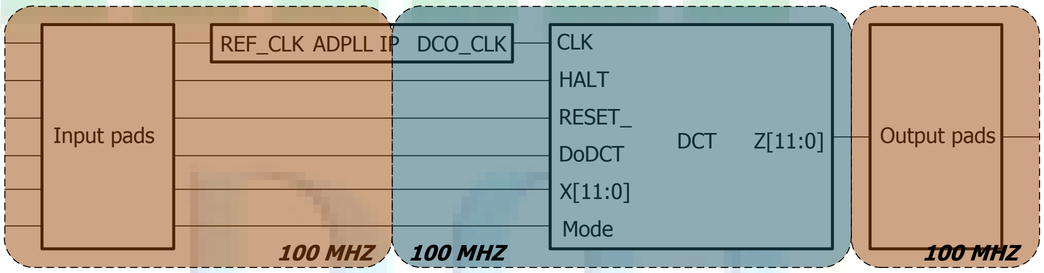
1. Change directory to lab 1
   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_1

2. Let us review design first.
   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_1/design

3. There is a core module as shown, our goals is to



   A. Package core module to a CHIP
   B. Use ADPLL IP to provide core clock
      i. Reference clock input: 100 MHz
      ii. ADPLL clock output: 1600 MHz
      iii. Core clock: 100 MHz



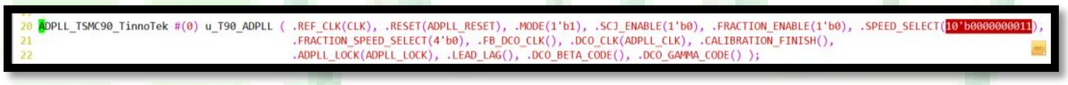
4. ADPLL SPEED_SELECT[9:0] pin configuration
   Reference clock: 100 MHz
   ADPLL clock: 1600 MHz

   If SPEED_SELECT[9] == 1’b0
   Then 1600 = 100 * (4 * SPEED_SELECT[8:0] + 4)
   Get SPEED_SELECT[8:0] = **9’d3**

   If SPEED_SELECT[9] == 1’b1
   Then 1600 = 100 * (4 * SPEED_SELECT[8:0] + 2)
   Can’t get the result

   Now, we can configure ADPLL IP as shown:

File name	CORE.v
Line	20



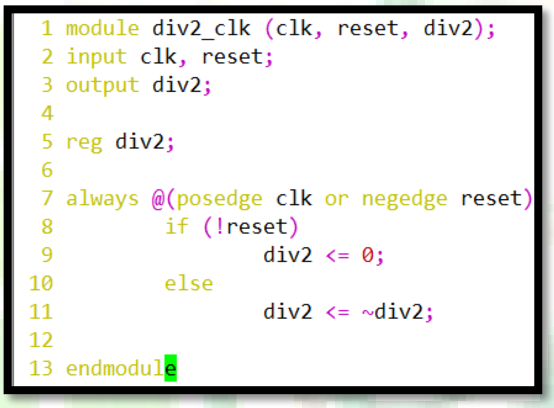
```
20 ADPLL_TSMC90_TinnoTek #(0) u_T90_ADPLL ( .REF_CLK(CLK), .RESET(ADPLL_RESET), .MODE(1'b1), .SCJ_ENABLE(1'b0), .FRACTION_ENABLE(1'b0), .SPEED_SELECT(10'b0000000011),
21                                           .FRACTION_SPEED_SELECT(4'b0), .FB_DCO_CLK(), .DCO_CLK(ADPLL_CLK), .CALIBRATION_FINISH(),
22                                           .ADPLL_LOCK(ADPLL_LOCK), .LEAD_LAG(), .DCO_BETA_CODE(), .DCO_GAMMA_CODE() );
```

5. Core clock is 100MHz, but ADPLL clock output is 1600MHz, therefore, we need frequency divider.

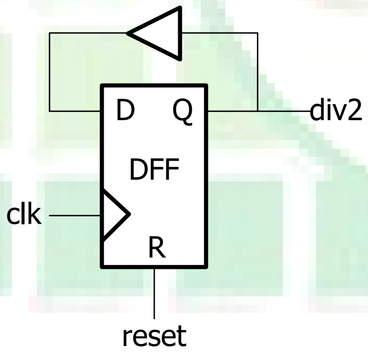
   Because 1600MHz clock frequency too high, so we partition frequency divider into two parts.

   A. First, use binary counter for high speed frequency divider.

File name	div2_clk.v

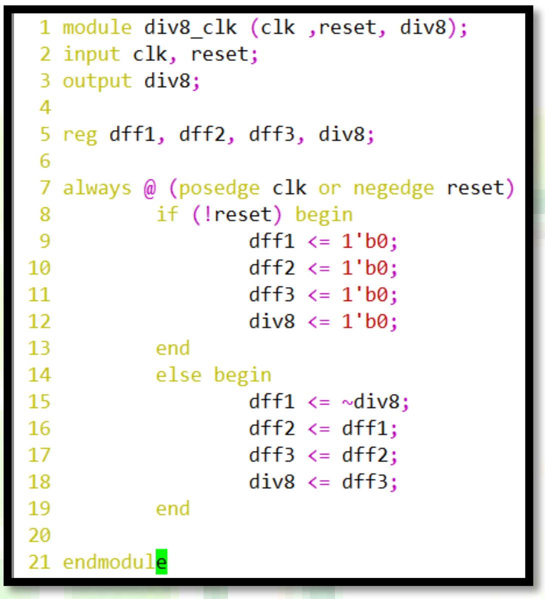


```verilog
module div2_clk (clk, reset, div2);
input clk, reset;
output div2;

reg div2;

always @(posedge clk or negedge reset)
        if (!reset)
                div2 <= 0;
        else
                div2 <= ~div2;

endmodule
```

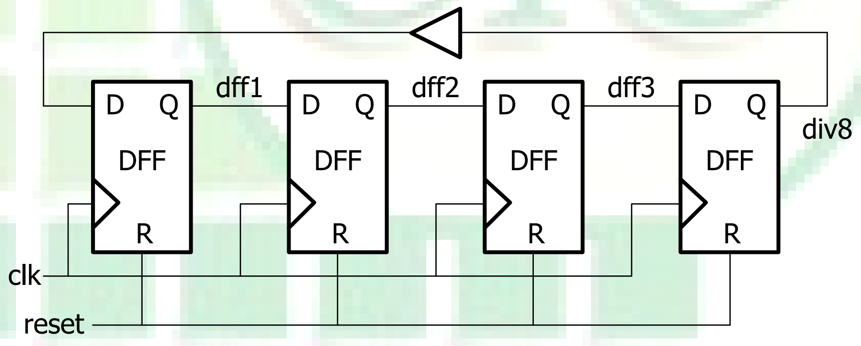


B. Second, use Jonson counter

File name	div8_clk.v

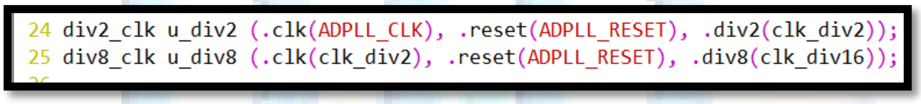


```verilog
module div8_clk (clk ,reset, div8);
input clk, reset;
output div8;

reg dff1, dff2, dff3, div8;

always @ (posedge clk or negedge reset)
        if (!reset) begin
                dff1 <= 1'b0;
                dff2 <= 1'b0;
                dff3 <= 1'b0;
                div8 <= 1'b0;
        end
        else begin
                dff1 <= ~div8;
                dff2 <= dff1;
                dff3 <= dff2;
                div8 <= dff3;
        end

endmodule
```

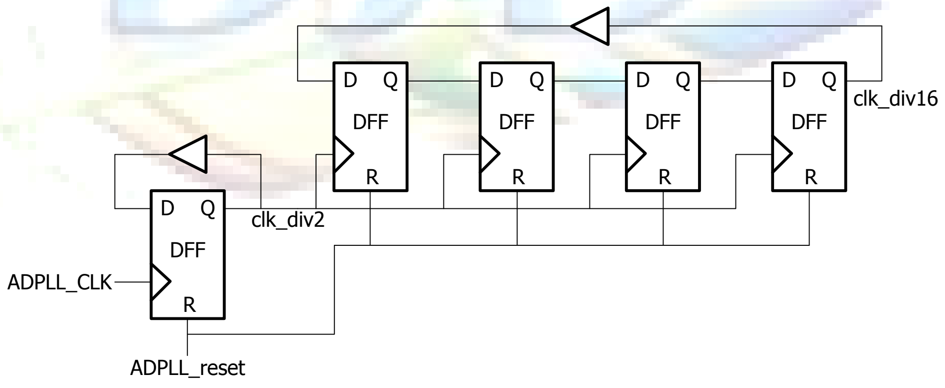


C. Finally, we can get a divide-by-16 frequency divider.

File name	CORE.v
Line number	24 and 25

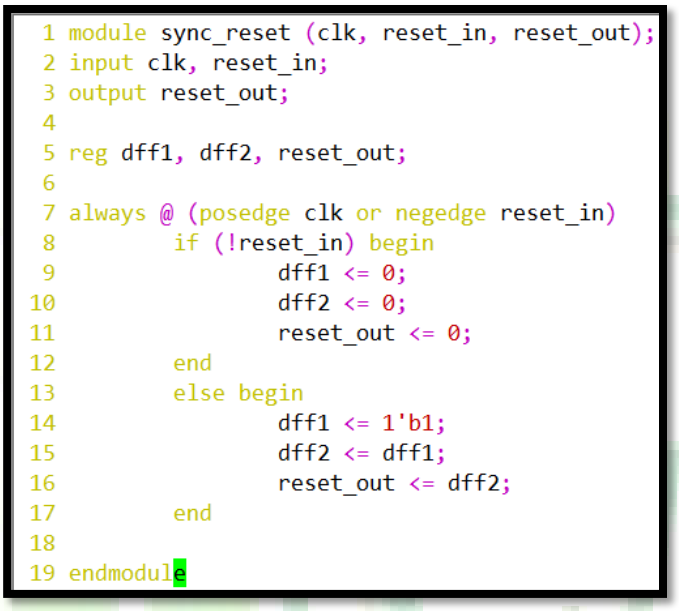


```verilog
24 div2_clk u_div2 (.clk(ADPLL_CLK), .reset(ADPLL_RESET), .div2(clk_div2));
25 div8_clk u_div8 (.clk(clk_div2), .reset(ADPLL_RESET), .div8(clk_div16));
```

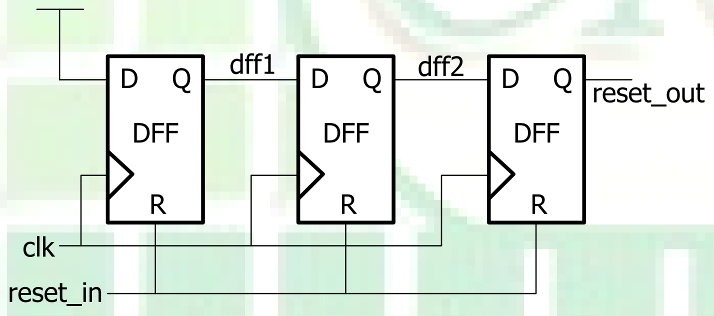


6. To prevent reset problems, we need reset synchronizer.

File name	sync_reset.v

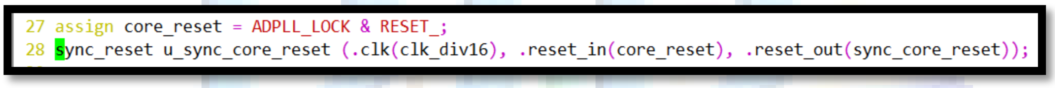


```verilog
module sync_reset (clk, reset_in, reset_out);
input clk, reset_in;
output reset_out;

reg dff1, dff2, reset_out;

always @ (posedge clk or negedge reset_in)
        if (!reset_in) begin
                dff1 <= 0;
                dff2 <= 0;
                reset_out <= 0;
        end
        else begin
                dff1 <= 1'b1;
                dff2 <= dff1;
                reset_out <= dff2;
        end

endmodule
```

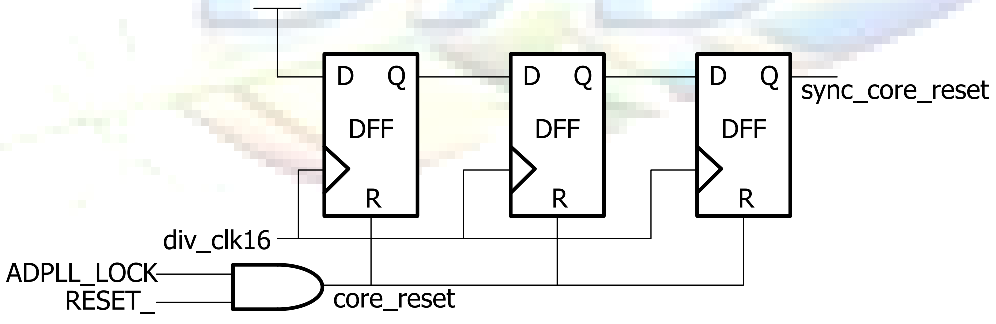


Because the core function MUST wait the ADPLL ADPLL_LOCK pin active high, we need an AND gate for core reset function.

File name	CORE.v
Line number	27 and 28



```verilog
27 assign core_reset = ADPLL_LOCK & RESET_;
28 sync_reset u_sync_core_reset (.clk(clk_div16), .reset_in(core_reset), .reset_out(sync_core_reset));
```

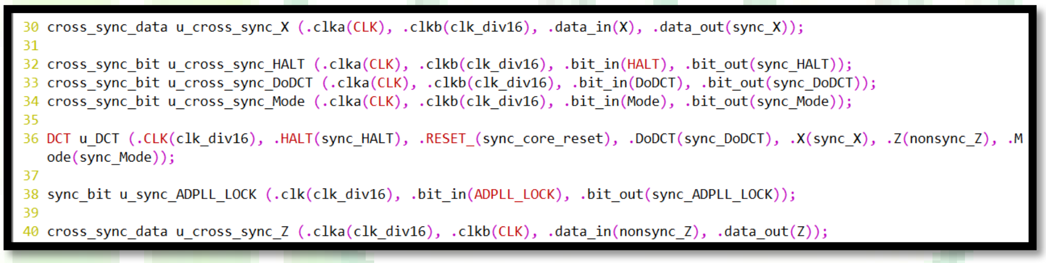


[Think about it]

Why frequency divider’s reset pin use “ADPLL_reset” instead of “sync_core_reset”?

7. Some of input/output ports are crossing two clock domains, so we need data synchronizers.

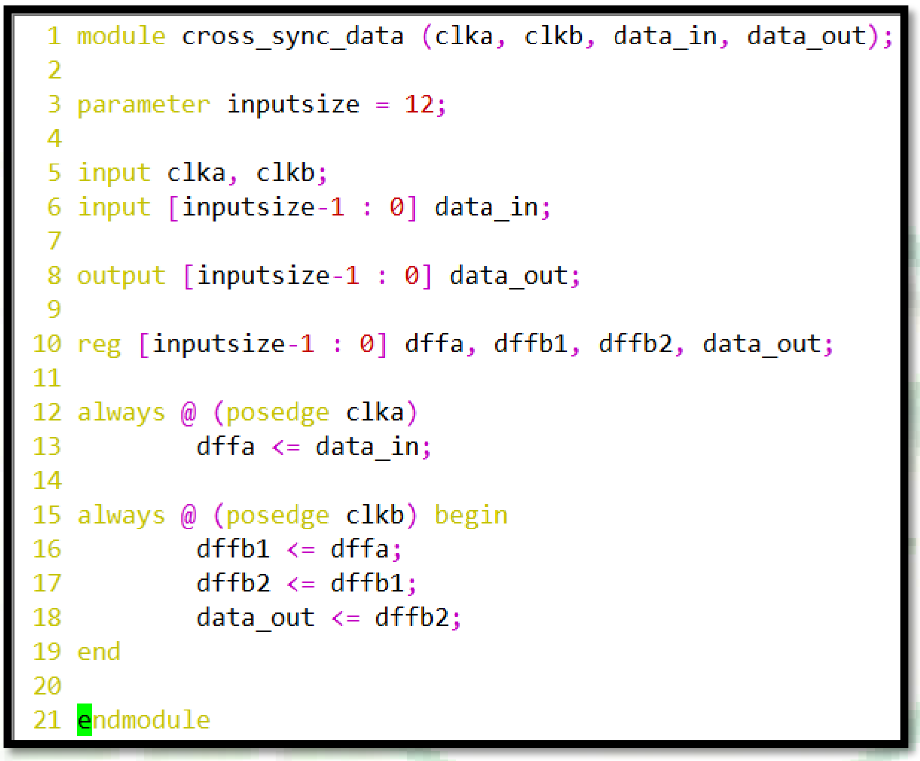
Port direction	Port name	SRC. Clock	DEST. Clock	Line number
Input	X	CLK	clk_div16	30
Input	HALT	CLK	clk_div16	32
Input	DoDCT	CLK	clk_div16	33
Input	Mode	CLK	clk_div16	34
Output	Z	clk_div16	CLK	40



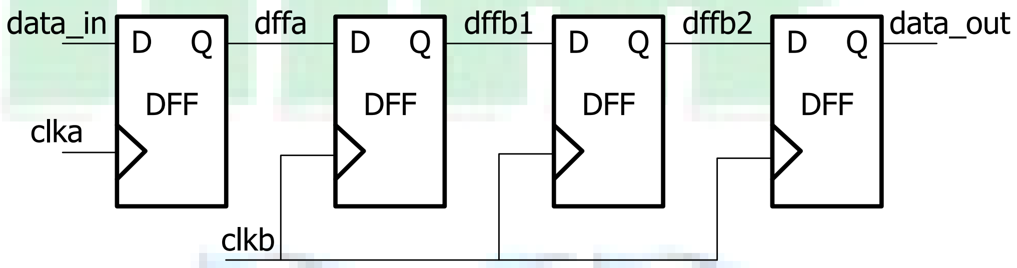
```
30 cross_sync_data u_cross_sync_X (.clka(CLK), .clkb(clk_div16), .data_in(X), .data_out(sync_X));
31
32 cross_sync_bit u_cross_sync_HALT (.clka(CLK), .clkb(clk_div16), .bit_in(HALT), .bit_out(sync_HALT));
33 cross_sync_bit u_cross_sync_DoDCT (.clka(CLK), .clkb(clk_div16), .bit_in(DoDCT), .bit_out(sync_DoDCT));
34 cross_sync_bit u_cross_sync_Mode (.clka(CLK), .clkb(clk_div16), .bit_in(Mode), .bit_out(sync_Mode));
35
36 DCT u_DCT (.CLK(clk_div16), .HALT(sync_HALT), .RESET_(sync_core_reset), .DoDCT(sync_DoDCT), .X(sync_X), .Z(nonsync_Z), .M
   ode(sync_Mode));
37
38 sync_bit u_sync_ADPLL_LOCK (.clk(clk_div16), .bit_in(ADPLL_LOCK), .bit_out(sync_ADPLL_LOCK));
39
40 cross_sync_data u_cross_sync_Z (.clka(clk_div16), .clkb(CLK), .data_in(nonsync_Z), .data_out(Z));
```

Multi/single bit data synchronizer

File name	cross_sync_bit.v cross_sync_data.v



```
module cross_sync_data (clka, clkb, data_in, data_out);

parameter inputsize = 12;

input clka, clkb;
input [inputsize-1 : 0] data_in;

output [inputsize-1 : 0] data_out;

reg [inputsize-1 : 0] dffa, dffb1, dffb2, data_out;

always @ (posedge clka)
        dffa <= data_in;

always @ (posedge clkb) begin
        dffb1 <= dffa;
        dffb2 <= dffb1;
        data_out <= dffb2;
end

endmodule
```

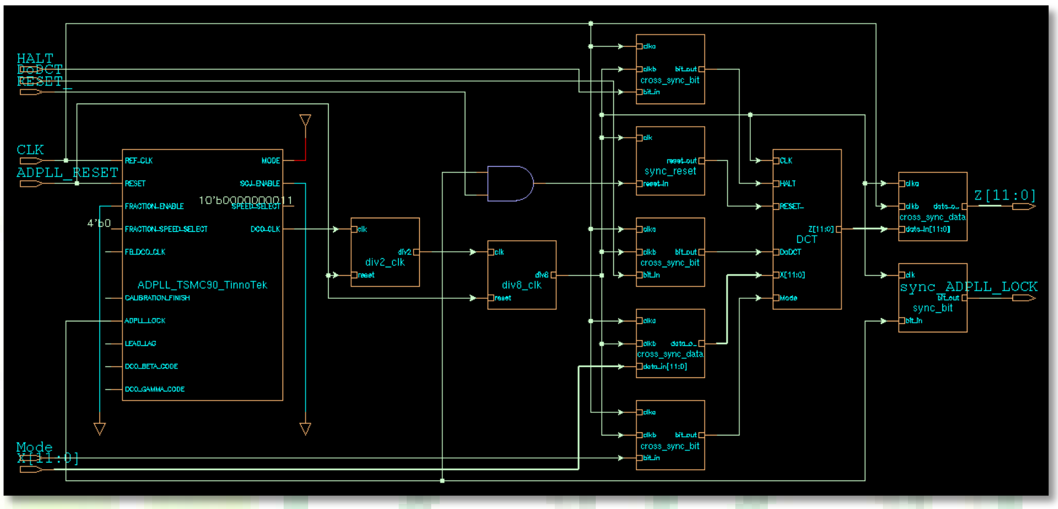


[Think about it]

Why output signal “ADPLL_LOCK” only sync with “div16_clk” clock?

8. Finally, put all designs together.

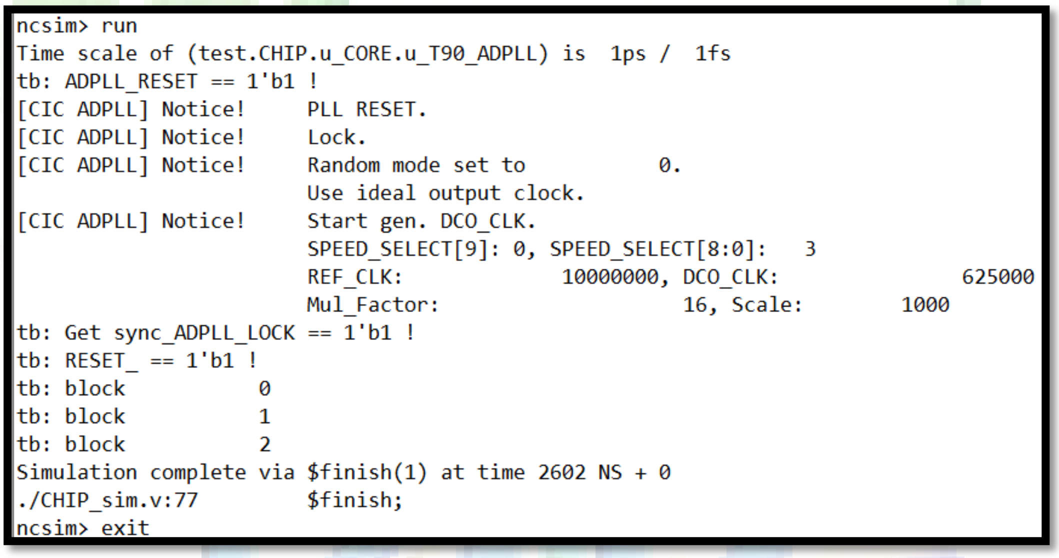
File name	CORE.v



9. And then running RTL simulation.

Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_1/sim_rtl

Unix% ncverilog –f vlog.f



```
ncsim> run
Time scale of (test.CHIP.u_CORE.u_T90_ADPLL) is  1ps /  1fs
tb: ADPLL_RESET == 1'b1 !
[CIC ADPLL] Notice!     PLL RESET.
[CIC ADPLL] Notice!     Lock.
[CIC ADPLL] Notice!     Random mode set to           0.
                        Use ideal output clock.
[CIC ADPLL] Notice!     Start gen. DCO_CLK.
                        SPEED_SELECT[9]: 0, SPEED_SELECT[8:0]:   3
                        REF_CLK:        10000000, DCO_CLK:              625000
                        Mul_Factor:               16, Scale:        1000
tb: Get sync_ADPLL_LOCK == 1'b1 !
tb: RESET_ == 1'b1 !
tb: block          0
tb: block          1
tb: block          2
Simulation complete via $finish(1) at time 2602 NS + 0
./CHIP_sim.v:77         $finish;
ncsim> exit
```

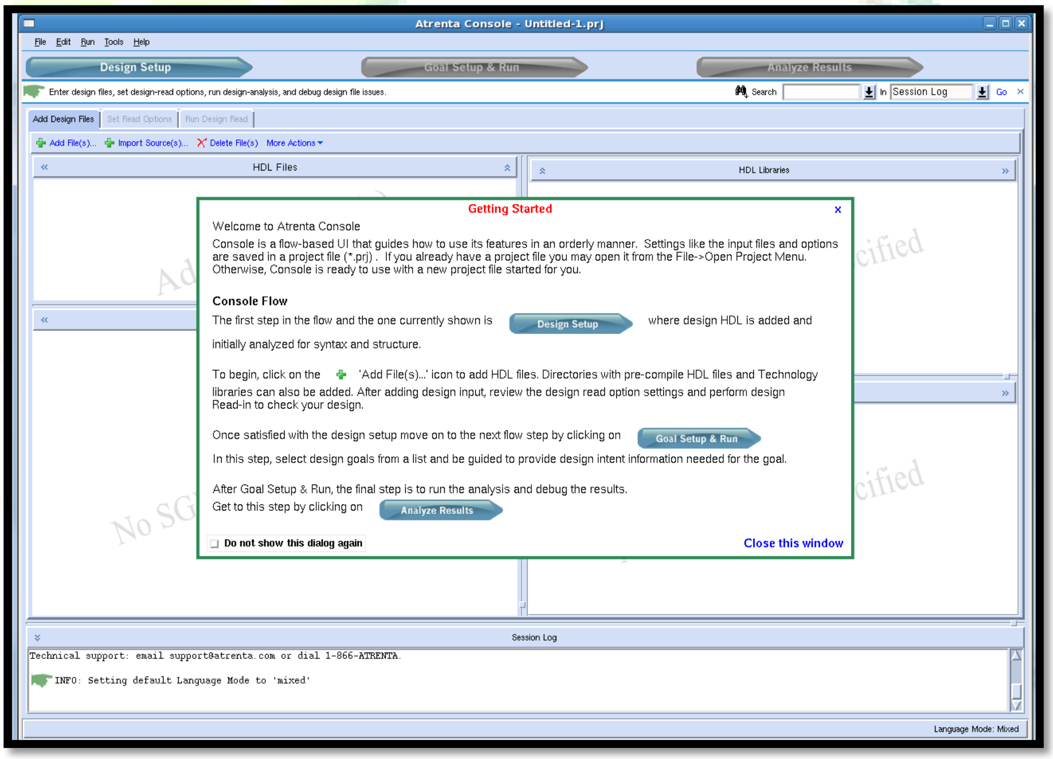
Lab 2 Case study: CDC CHECK

1. Change directory to lab 2

   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_2

2. Open the SpyGlass in interactive mode

   Unix% spyglass



Enter design files, set design-read options, run design-analysis, and debug design file issues.

Getting Started

Welcome to Atrenta Console

Console is a flow-based UI that guides how to use its features in an orderly manner. Settings like the input files and options are saved in a project file (*.prj) . If you already have a project file you may open it from the File->Open Project Menu. Otherwise, Console is ready to use with a new project file started for you.

Console Flow

The first step in the flow and the one currently shown is Design Setup where design HDL is added and initially analyzed for syntax and structure.

To begin, click on the 'Add File(s)...' icon to add HDL files. Directories with pre-compile HDL files and Technology libraries can also be added. After adding design input, review the design read option settings and perform design Read-in to check your design.

Once satisfied with the design setup move on to the next flow step by clicking on Goal Setup & Run

In this step, select design goals from a list and be guided to provide design intent information needed for the goal.

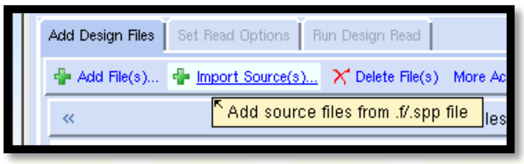
After Goal Setup & Run, the final step is to run the analysis and debug the results.
Get to this step by clicking on Analyze Results

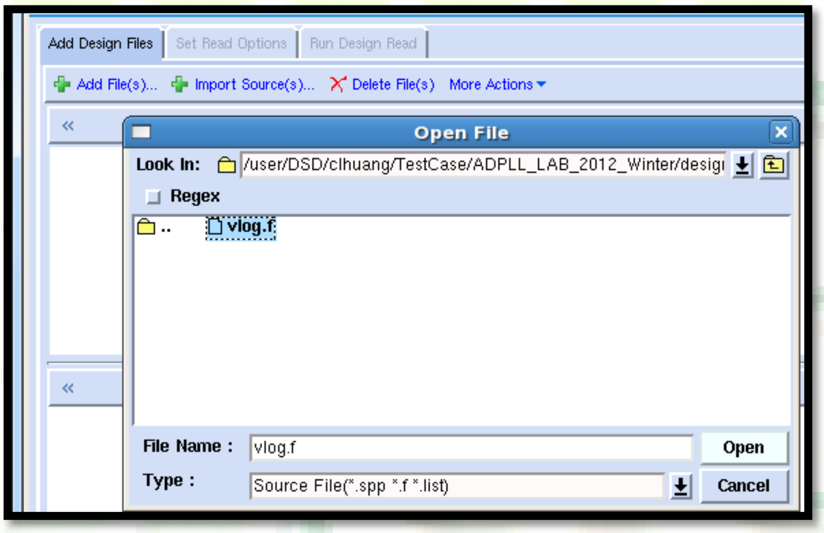
Do not show this dialog again

Close this window

```
Technical support: email support@atrenta.com or dial 1-866-ATRENTA.
INFO: Setting default Language Mode to 'mixed'
```

3. Reading design list

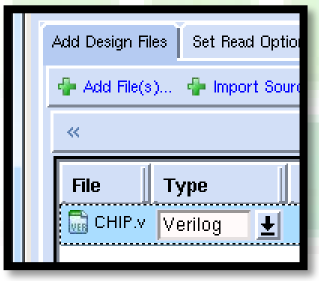




Look In: /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/desig

File Name : vlog.f

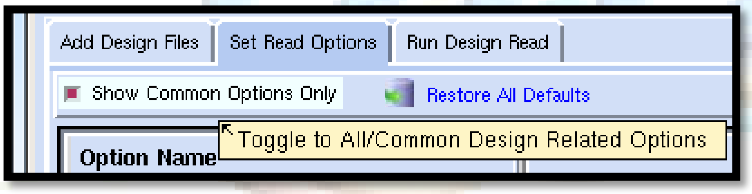
Type : Source File(*.spp *.f *.list)



File	Type
CHIP.v	Verilog

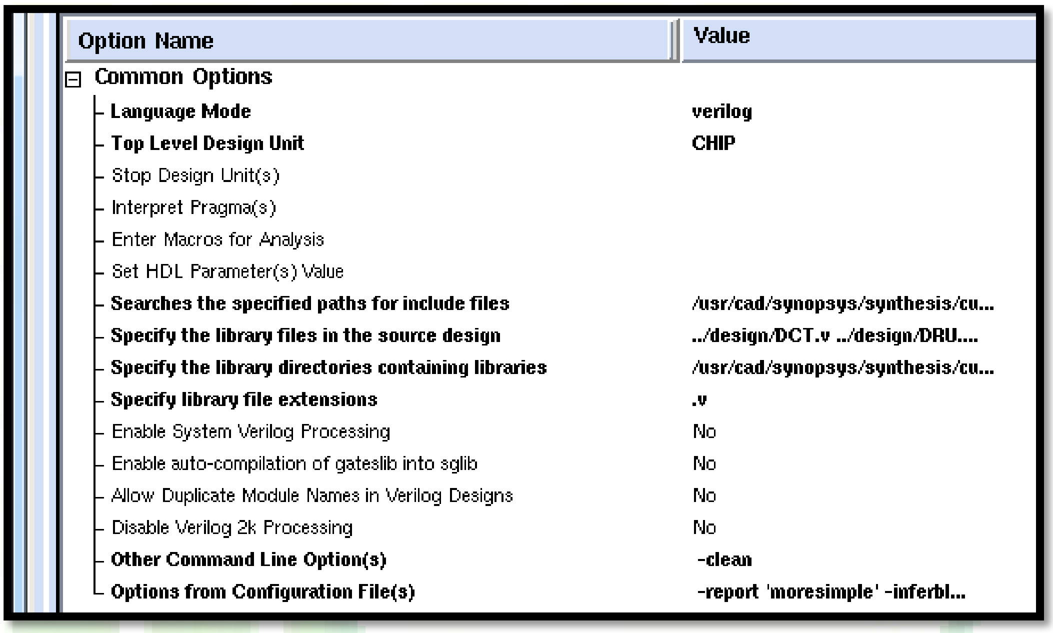
4. Set read options

Un-check “Show Common Option Only”



Setting common options

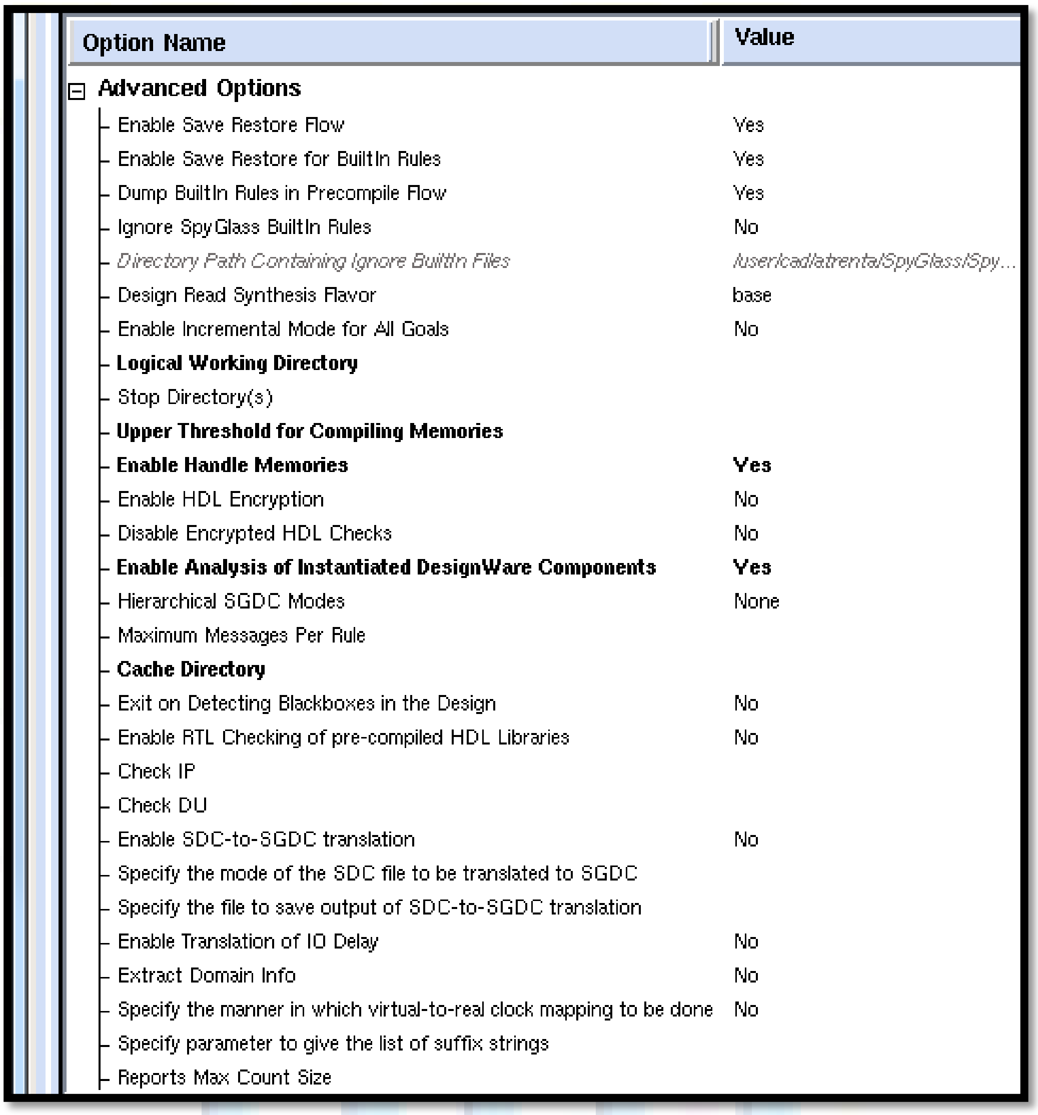
Language Mode	**verilog**
Top Level Design Unit	**CHIP**
Others	Default



Option Name	Value
⊟ Common Options	
Language Mode	**verilog**
Top Level Design Unit	**CHIP**
Stop Design Unit(s)	
Interpret Pragma(s)	
Enter Macros for Analysis	
Set HDL Parameter(s) Value	
Searches the specified paths for include files	**/usr/cad/synopsys/synthesis/cu...**
Specify the library files in the source design	**../design/DCT.v ../design/DRU....**
Specify the library directories containing libraries	**/usr/cad/synopsys/synthesis/cu...**
Specify library file extensions	**.v**
Enable System Verilog Processing	No
Enable auto-compilation of gateslib into sglib	No
Allow Duplicate Module Names in Verilog Designs	No
Disable Verilog 2k Processing	No
Other Command Line Option(s)	**-clean**
Options from Configuration File(s)	**-report 'moresimple' -inferbl...**

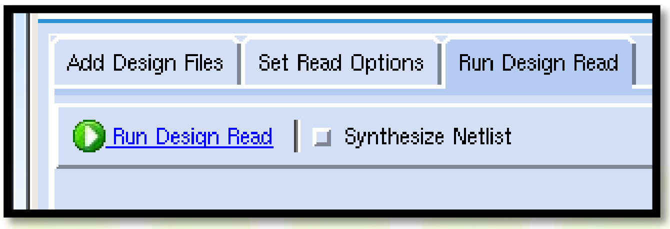
Setting advanced options

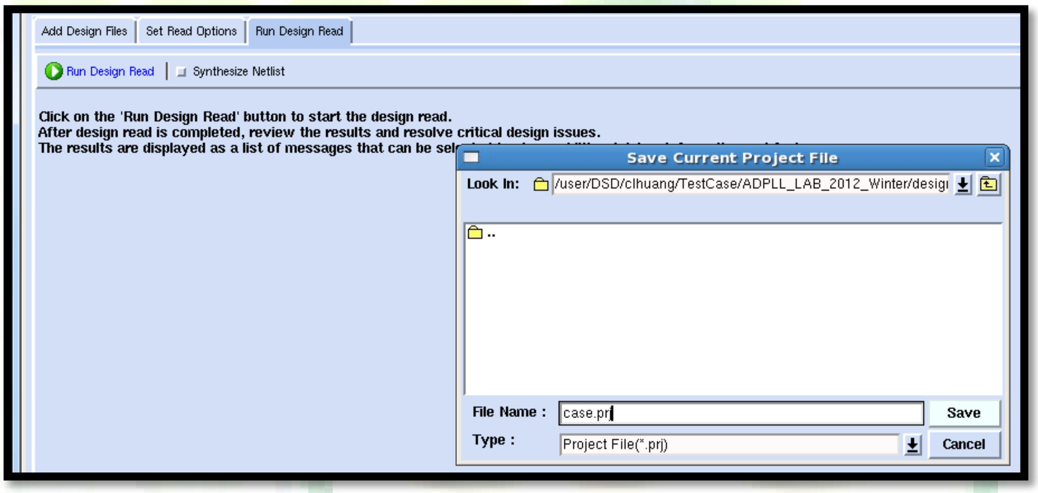
Enable Handle Memories	**Yes**
Enable Analysis of Instantiated DesignWare Components	**Yes**
Others	Default



Option Name	Value
Advanced Options	
Enable Save Restore Flow	Yes
Enable Save Restore for BuiltIn Rules	Yes
Dump BuiltIn Rules in Precompile Flow	Yes
Ignore SpyGlass BuiltIn Rules	No
Directory Path Containing Ignore BuiltIn Files	*/usr/cad/atrenta/SpyGlass/Spy...*
Design Read Synthesis Flavor	base
Enable Incremental Mode for All Goals	No
Logical Working Directory	
Stop Directory(s)	
Upper Threshold for Compiling Memories	
Enable Handle Memories	**Yes**
Enable HDL Encryption	No
Disable Encrypted HDL Checks	No
Enable Analysis of Instantiated DesignWare Components	**Yes**
Hierarchical SGDC Modes	None
Maximum Messages Per Rule	
Cache Directory	
Exit on Detecting Blackboxes in the Design	No
Enable RTL Checking of pre-compiled HDL Libraries	No
Check IP	
Check DU	
Enable SDC-to-SGDC translation	No
Specify the mode of the SDC file to be translated to SGDC	
Specify the file to save output of SDC-to-SGDC translation	
Enable Translation of IO Delay	No
Extract Domain Info	No
Specify the manner in which virtual-to-real clock mapping to be done	No
Specify parameter to give the list of suffix strings	
Reports Max Count Size	

5. Run Design Read and save project “case.prj”





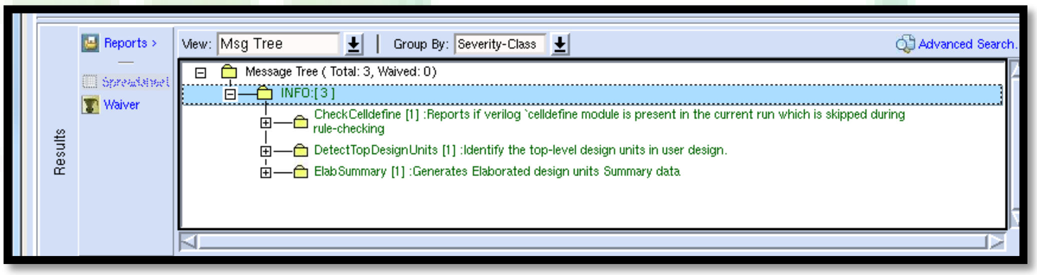
Click on the 'Run Design Read' button to start the design read.
After design read is completed, review the results and resolve critical design issues.
The results are displayed as a list of messages that can be sel

Look In: /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/desigı

File Name : case.prj

Type : Project File(*.prj)

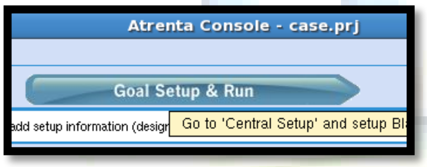
6. Check “Msg Tree”, make sure everything is fine.



View: Msg Tree | Group By: Severity-Class

```
Message Tree ( Total: 3, Waived: 0)
  INFO:[ 3 ]
    CheckCelldefine [1] :Reports if verilog `celldefine module is present in the current run which is skipped during
    rule-checking
    DetectTopDesignUnits [1] :Identify the top-level design units in user design.
    ElabSummary [1] :Generates Elaborated design units Summary data
```

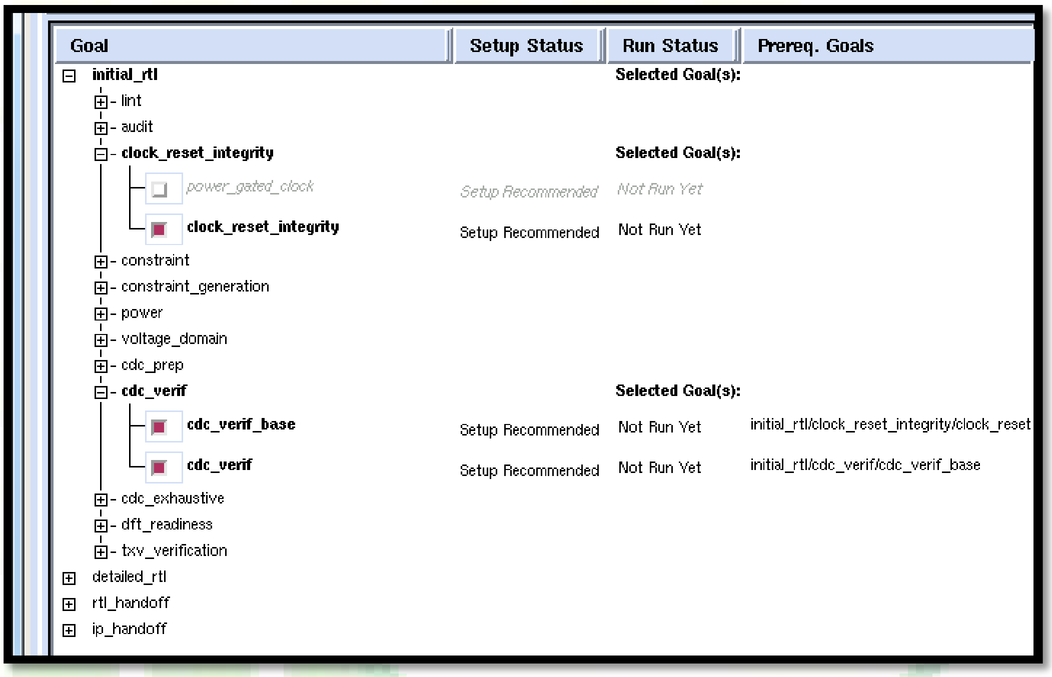
7. Switch to “Goal Setup & Run”



add setup information (design Go to 'Central Setup' and setup Bl

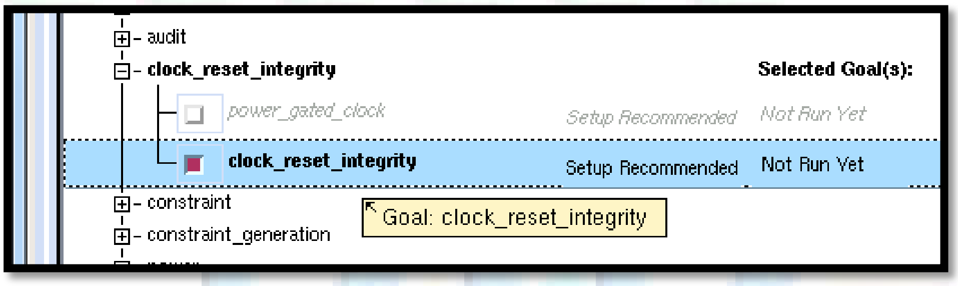
8. Select goals

Initial_rtl	clock_reset_integrity	clock_reset_integrity
Initial_rtl	cdc_verif	cdc_verif_base
Initial_rtl	cdc_verif	cdc_verif



Goal	Setup Status	Run Status	Prereq. Goals
initial_rtl		Selected Goal(s):	
- lint			
- audit			
- clock_reset_integrity		Selected Goal(s):	
power_gated_clock	Setup Recommended	Not Run Yet	
clock_reset_integrity	Setup Recommended	Not Run Yet	
- constraint			
- constraint_generation			
- power			
- voltage_domain			
- cdc_prep			
- cdc_verif		Selected Goal(s):	
cdc_verif_base	Setup Recommended	Not Run Yet	initial_rtl/clock_reset_integrity/clock_reset
cdc_verif	Setup Recommended	Not Run Yet	initial_rtl/cdc_verif/cdc_verif_base
- cdc_exhaustive			
- dft_readiness			
- txv_verification			
detailed_rtl			
rtl_handoff			
ip_handoff			

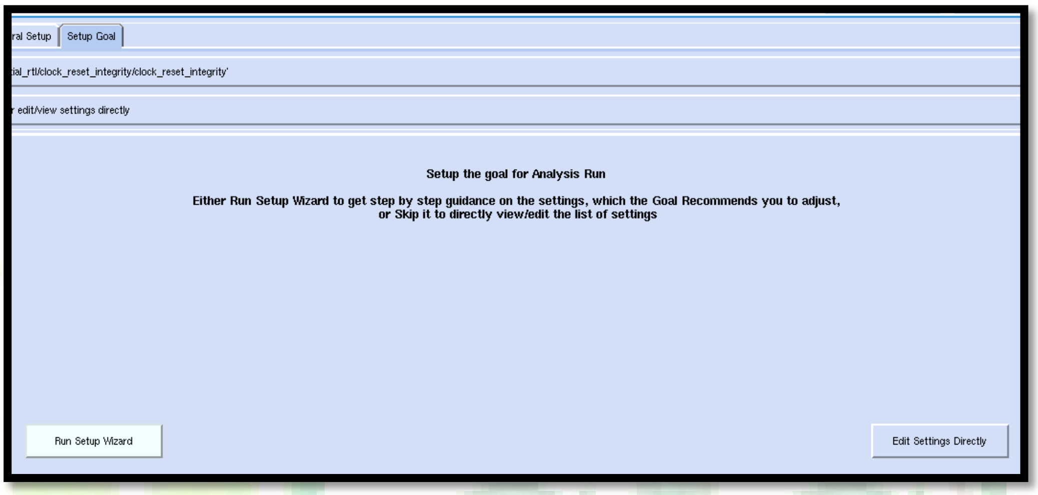
9. Highlight the goal "clock_reset_integrity" by right-click and then perform "Setup Goal"



- audit		
- clock_reset_integrity		Selected Goal(s):
power_gated_clock	Setup Recommended	Not Run Yet
clock_reset_integrity	Setup Recommended	Not Run Yet
- constraint		
- constraint_generation		

Goal: clock_reset_integrity

10. Switch function tab to “Setup Goal” and run “Run Setup Wizard”



ial_rtl/clock_reset_integrity/clock_reset_integrity'

r edit/view settings directly

Setup the goal for Analysis Run

Either Run Setup Wizard to get step by step guidance on the settings, which the Goal Recommends you to adjust,
or Skip it to directly view/edit the list of settings

11. Click “Next” to perform “Setup Preference”.

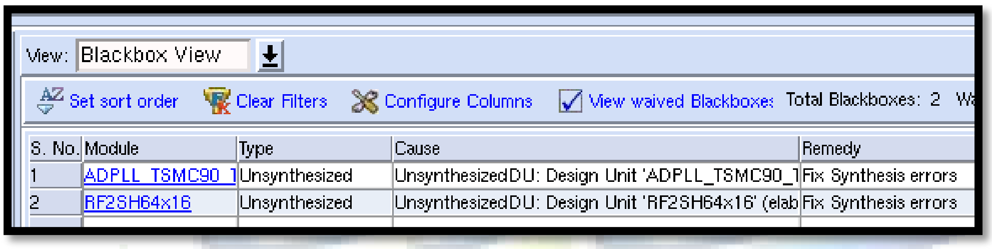
12. Click “No” to skip advanced setup.

Do you want to perform advanced setup?	**No**

13. Let Spyglass identify everything.

Do you have any SGDC files?	No
Do you want to import constraints from SDC files	No
Identify potential clocks used in the design?	**Yes**

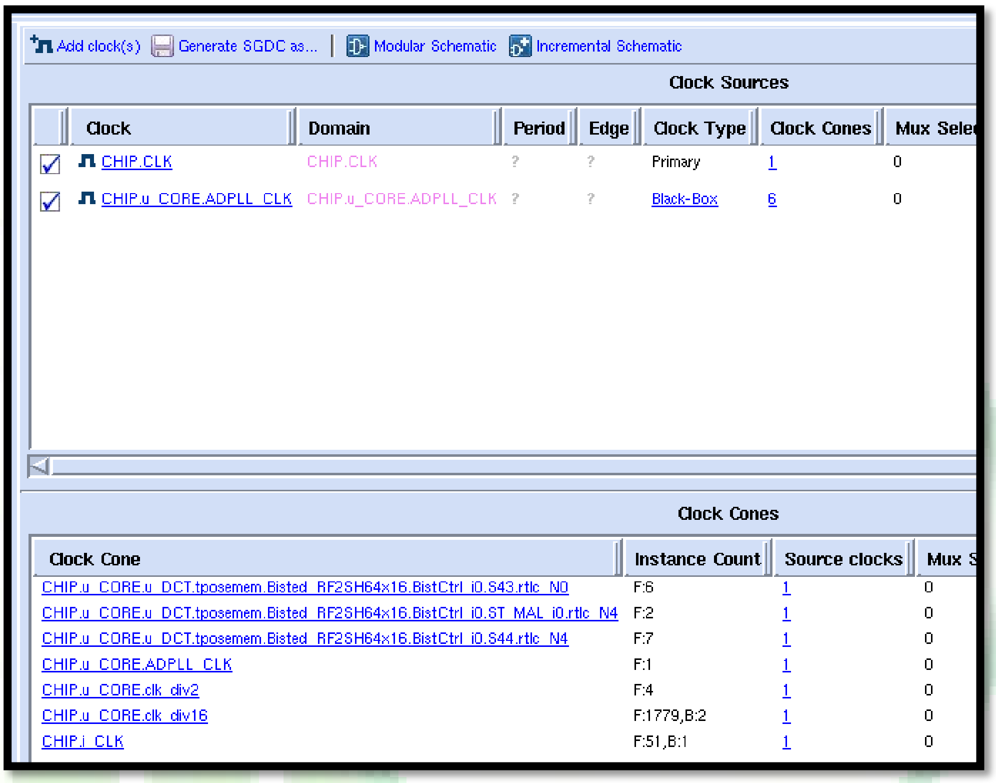
14. Click “Next” to perform “Resolve Blackboxes”



S. No.	Module	Type	Cause	Remedy
1	ADPLL_TSMC90_1	Unsynthesized	UnsynthesizedDU: Design Unit 'ADPLL_TSMC90_1	Fix Synthesis errors
2	RF2SH64x16	Unsynthesized	UnsynthesizedDU: Design Unit 'RF2SH64x16' (elab	Fix Synthesis errors

15. Click "Next" and show clock trees.

Show clock trees and finalize clock definition interactively?	**Yes**



Add clock(s) | Generate SGDC as ... | Modular Schematic | Incremental Schematic

Clock Sources

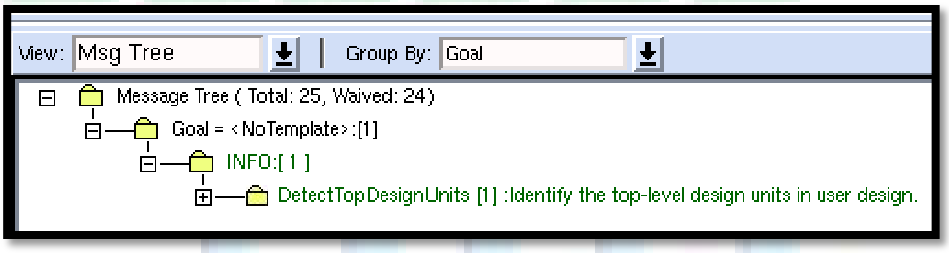
	Clock	Domain	Period	Edge	Clock Type	Clock Cones	Mux Sele
✓	CHIP.CLK	CHIP.CLK	?	?	Primary	1	0
✓	CHIP.u_CORE.ADPLL_CLK	CHIP.u_CORE.ADPLL_CLK	?	?	Black-Box	6	0

Clock Cones

Clock Cone	Instance Count	Source clocks	Mux S
CHIP.u_CORE.u_DCT.tposemem.Bisted_RF2SH64x16.BistCtrl_i0.S43.rtlc_N0	F:6	1	0
CHIP.u_CORE.u_DCT.tposemem.Bisted_RF2SH64x16.BistCtrl_i0.ST_MAL_i0.rtlc_N4	F:2	1	0
CHIP.u_CORE.u_DCT.tposemem.Bisted_RF2SH64x16.BistCtrl_i0.S44.rtlc_N4	F:7	1	0
CHIP.u_CORE.ADPLL_CLK	F:1	1	0
CHIP.u_CORE.clk_div2	F:4	1	0
CHIP.u_CORE.clk_div16	F:1779,B:2	1	0
CHIP.i_CLK	F:51,B:1	1	0

16. Click "Next" to verify clock setup.

Verify clock setup?	**Yes**

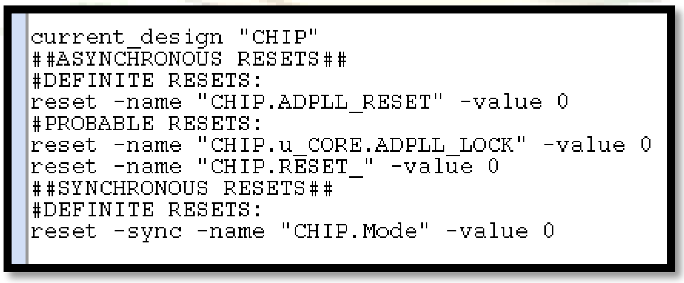


View: Msg Tree | Group By: Goal

```
Message Tree ( Total: 25, Waived: 24 )
  Goal = <NoTemplate>:[1]
    INFO:[ 1 ]
      DetectTopDesignUnits [1] :Identify the top-level design units in user design.
```

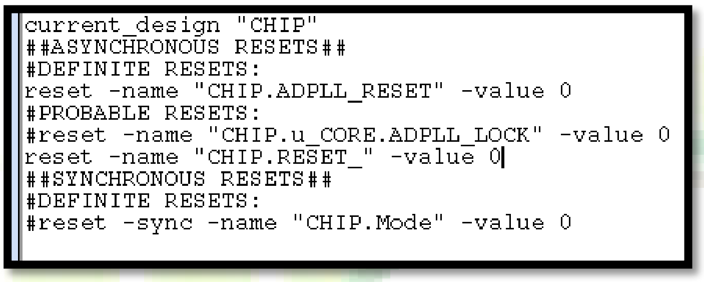
17. Click "Next" to setup reset signals

Edit and complete reset constraints?	**Yes**



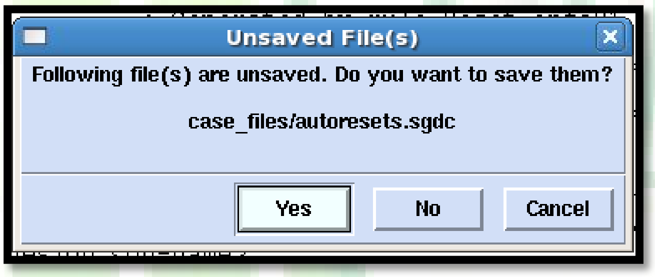
```
current_design "CHIP"
##ASYNCHRONOUS RESETS##
#DEFINITE RESETS:
reset -name "CHIP.ADPLL_RESET" -value 0
#PROBABLE RESETS:
reset -name "CHIP.u_CORE.ADPLL_LOCK" -value 0
reset -name "CHIP.RESET_" -value 0
##SYNCHRONOUS RESETS##
#DEFINITE RESETS:
reset -sync -name "CHIP.Mode" -value 0
```

Because the signal “CHIP.Mode” and “CHIP.u_CORE.ADPLL_LOCK” act like reset behavior, Spyglass identifies them. In this case, we can’t need these two signals, so we comment them by “#”.



```
current_design "CHIP"
##ASYNCHRONOUS RESETS##
#DEFINITE RESETS:
reset -name "CHIP.ADPLL_RESET" -value 0
#PROBABLE RESETS:
#reset -name "CHIP.u_CORE.ADPLL_LOCK" -value 0
reset -name "CHIP.RESET_" -value 0
##SYNCHRONOUS RESETS##
#DEFINITE RESETS:
#reset -sync -name "CHIP.Mode" -value 0
```

Click “Next” to save file.

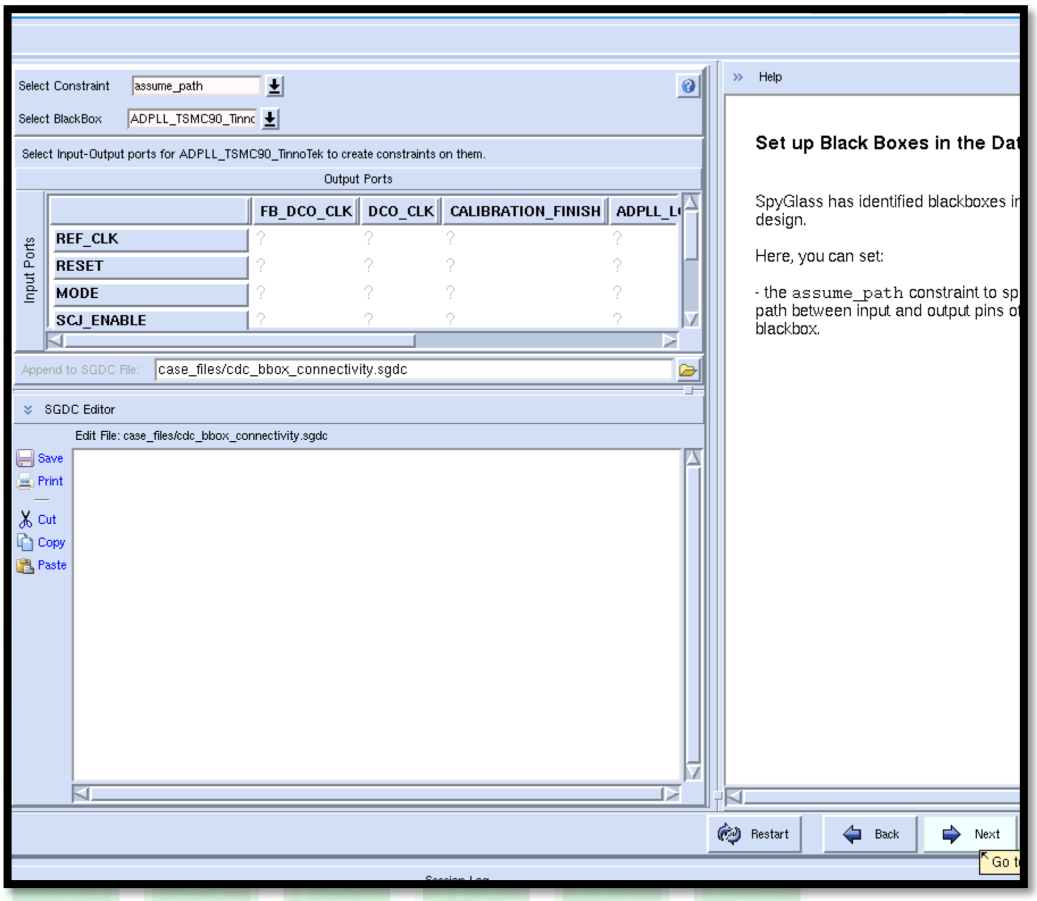


Unsaved File(s)

Following file(s) are unsaved. Do you want to save them?

case_files/autoresets.sgdc

18. Because all of ADPLL data input/output can treat as false paths, except ADPLL_LOCK signal. But this “ADPLL_LOCK” is asynchronous signal, so we can skip “Set up Black Boxes in the Data Path” by clicking “Next”.


Select Constraint assume_path

Select BlackBox ADPLL_TSMC90_Tinnc

Select Input-Output ports for ADPLL_TSMC90_TinnoTek to create constraints on them.

Output Ports

	FB_DCO_CLK	DCO_CLK	CALIBRATION_FINISH	ADPLL_L
REF_CLK	?	?	?	?
RESET	?	?	?	?
MODE	?	?	?	?
SCJ_ENABLE	?	?	?	?

Append to SGDC File: case_files/cdc_bbox_connectivity.sgdc

SGDC Editor

Edit File: case_files/cdc_bbox_connectivity.sgdc

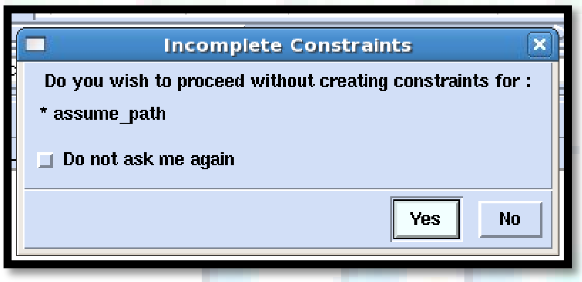
Help

Set up Black Boxes in the Dat

SpyGlass has identified blackboxes i design.

Here, you can set:

- the `assume_path` constraint to sp path between input and output pins o blackbox.

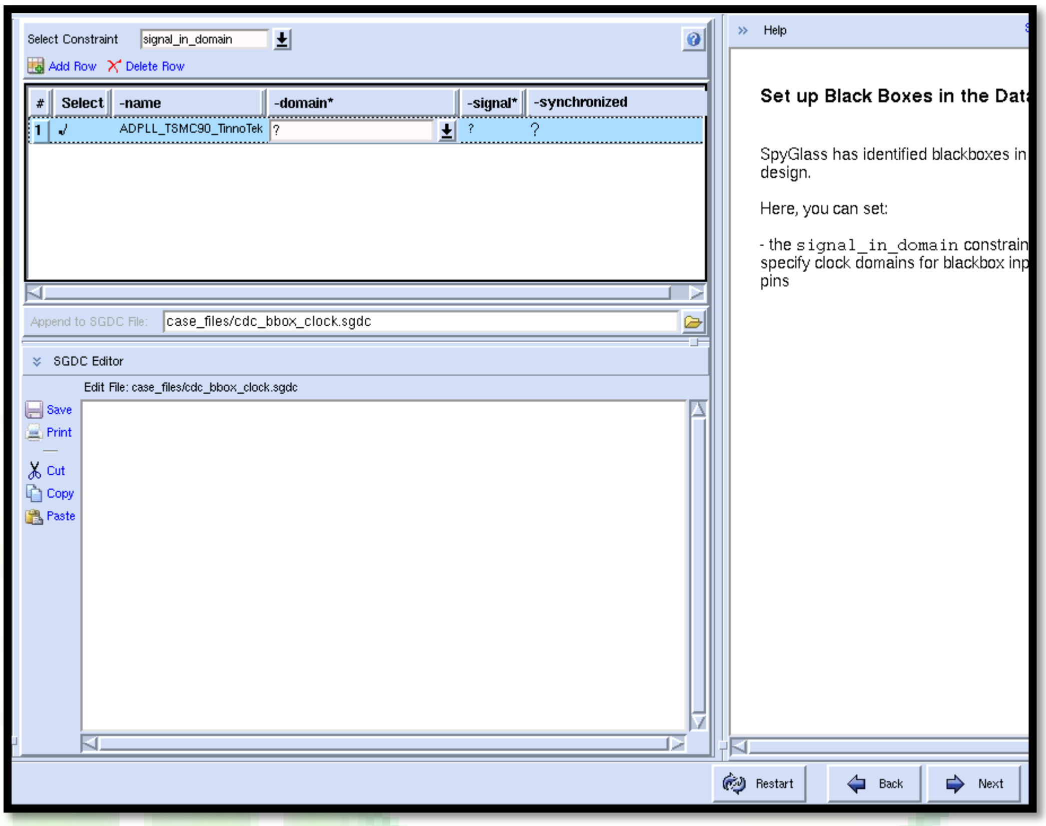

Incomplete Constraints

Do you wish to proceed without creating constraints for :

* assume_path

☐ Do not ask me again

Also, we can skip “signal_in_domain” SGDC setting by clicking “Next”.



#	Select	-name	-domain*	-signal*	-synchronized
1	✓	ADPLL_TSMC90_TinnoTek	?	?	?

case_files/cdc_bbox_clock.sgdc

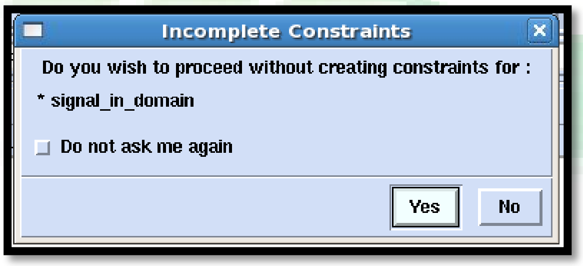
Edit File: case_files/cdc_bbox_clock.sgdc

Set up Black Boxes in the Dat

SpyGlass has identified blackboxes in design.

Here, you can set:

- the signal_in_domain constrain specify clock domains for blackbox inp pins



Do you wish to proceed without creating constraints for :

* signal_in_domain

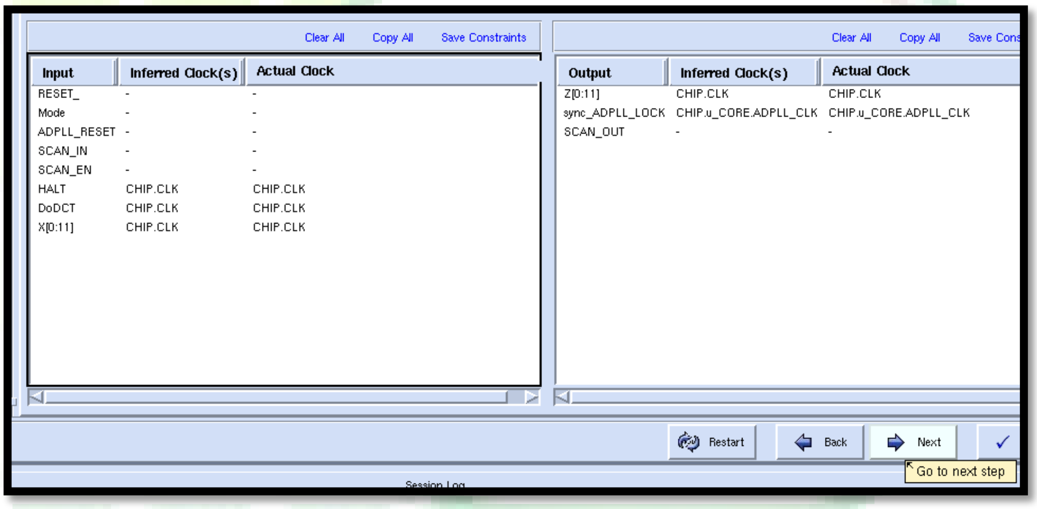
Do not ask me again

19. Click “Yes” to edit IO domains.

Edit and complete IO Domains	**Yes**

Use “Copy all” to setting input/output “Actual Clock”

Input	HALT	CHIP.CLK	CHIP.CLK
Input	DoDCT	CHIP.CLK	CHIP.CLK
Input	X[0:11]	CHIP.CLK	CHIP.CLK
Input	Others	NA	NA
Output	Z[0:11]	CHIP.CLK	CHIP.CLK
Output	Sync_ADPLL_LOCK	CHIP.u_CORE.ADPLL_CLK	CHIP.u_CORE_ADPLL_CLK
Output	Others	NA	NA

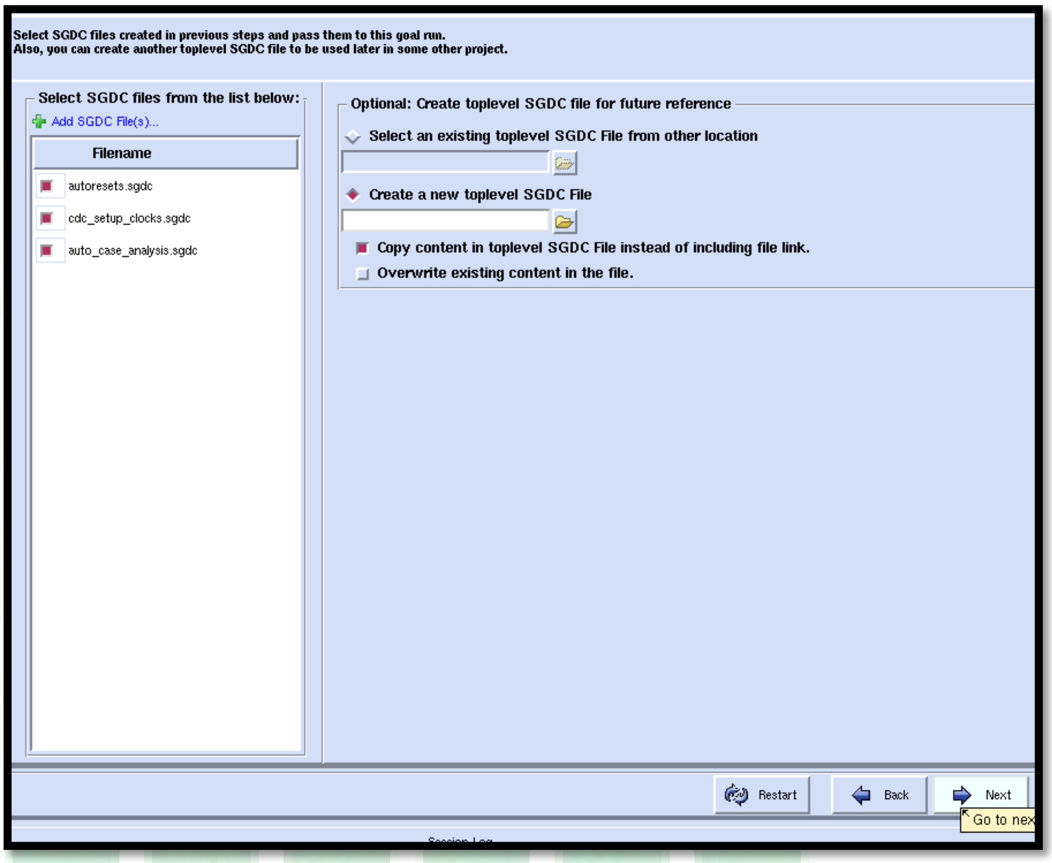


Input	Inferred Clock(s)	Actual Clock
RESET_	-	-
Mode	-	-
ADPLL_RESET	-	-
SCAN_IN	-	-
SCAN_EN	-	-
HALT	CHIP.CLK	CHIP.CLK
DoDCT	CHIP.CLK	CHIP.CLK
X[0:11]	CHIP.CLK	CHIP.CLK

Output	Inferred Clock(s)	Actual Clock
Z[0:11]	CHIP.CLK	CHIP.CLK
sync_ADPLL_LOCK	CHIP.u_CORE.ADPLL_CLK	CHIP.u_CORE.ADPLL_CLK
SCAN_OUT	-	-

20. Save SGDC file

Copy content in toplevel SGDC file instead of including file link	**Check**
Others	Default


Select SGDC files created in previous steps and pass them to this goal run.
Also, you can create another toplevel SGDC file to be used later in some other project.

Select SGDC files from the list below:

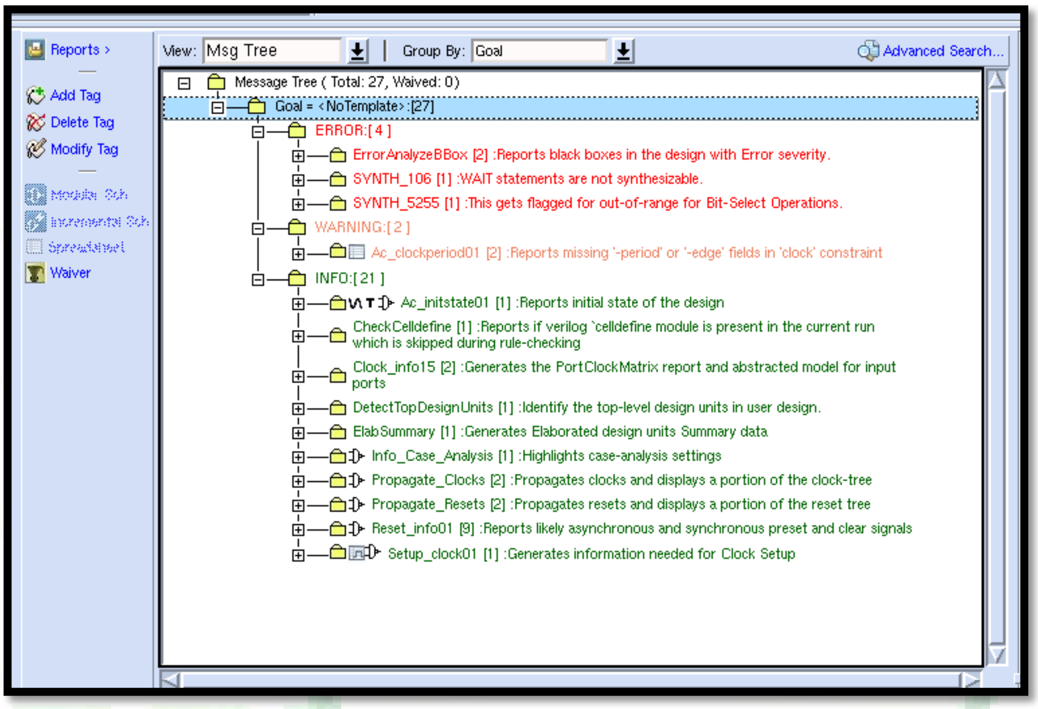
Filename
🟥 autoresets.sgdc
🟥 cdc_setup_clocks.sgdc
🟥 auto_case_analysis.sgdc

Optional: Create toplevel SGDC file for future reference

- ◇ Select an existing toplevel SGDC File from other location
- ◆ Create a new toplevel SGDC File
- 🟥 Copy content in toplevel SGDC File instead of including file link.
- ☐ Overwrite existing content in the file.

21. Click “Next” to perform “Check issues with Setup for Formal Verification”

Check setup now?	**Yes**

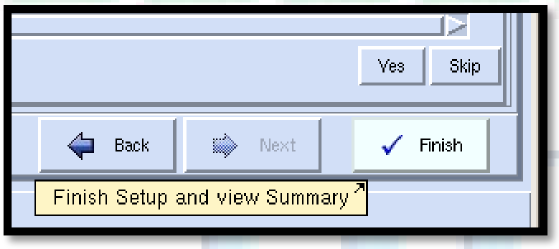


Message Tree (Total: 27, Waived: 0)
Goal = <NoTemplate>:[27]
ERROR:[4]
ErrorAnalyzeBBox [2] :Reports black boxes in the design with Error severity.
SYNTH_106 [1] :WAIT statements are not synthesizable.
SYNTH_5255 [1] :This gets flagged for out-of-range for Bit-Select Operations.
WARNING:[2]
Ac_clockperiod01 [2] :Reports missing '-period' or '-edge' fields in 'clock' constraint
INFO:[21]
Ac_initstate01 [1] :Reports initial state of the design
CheckCelldefine [1] :Reports if verilog `celldefine module is present in the current run which is skipped during rule-checking
Clock_info15 [2] :Generates the PortClockMatrix report and abstracted model for input ports
DetectTopDesignUnits [1] :Identify the top-level design units in user design.
ElabSummary [1] :Generates Elaborated design units Summary data
Info_Case_Analysis [1] :Highlights case-analysis settings
Propagate_Clocks [2] :Propagates clocks and displays a portion of the clock-tree
Propagate_Resets [2] :Propagates resets and displays a portion of the reset tree
Reset_info01 [9] :Reports likely asynchronous and synchronous preset and clear signals
Setup_clock01 [1] :Generates information needed for Clock Setup

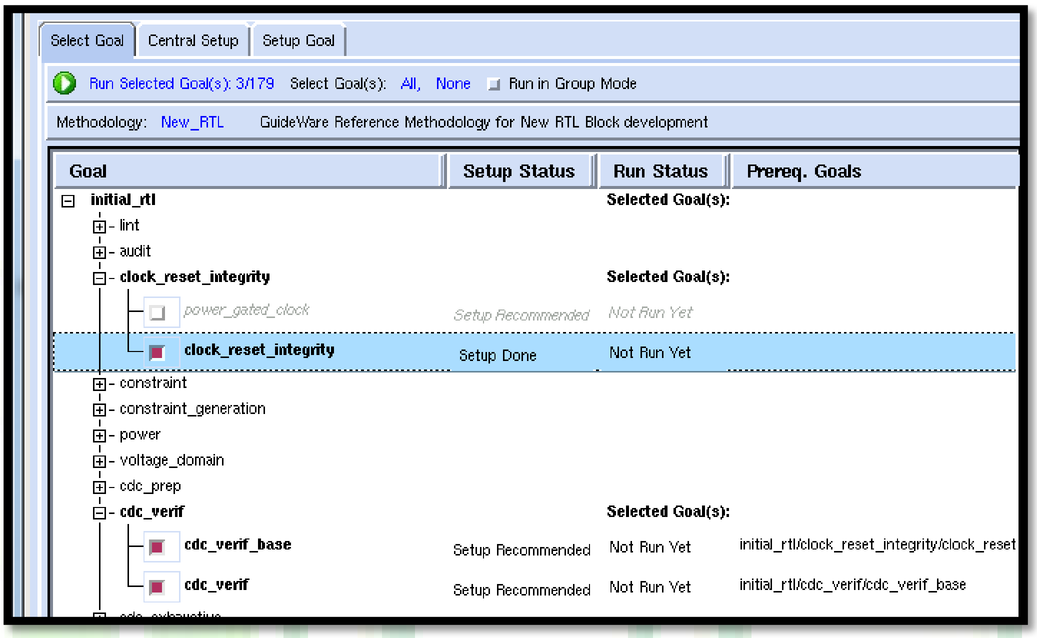
[Think about it]

Why we can ignore all the errors?

22. Click “Next” and finish setup.



23. Switch function tab back to "Select Goal"

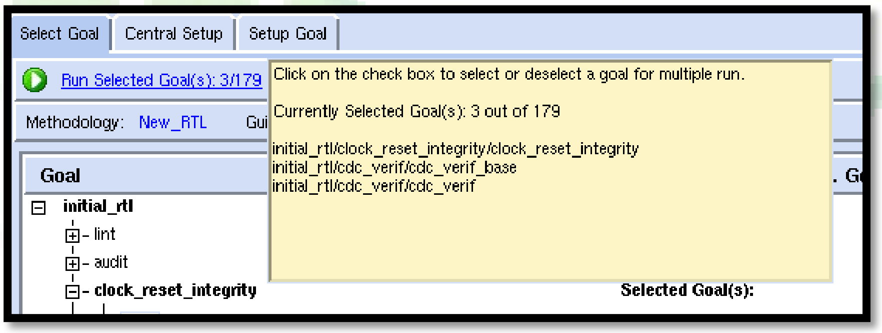

Select Goal | Central Setup | Setup Goal

Run Selected Goal(s): 3/179   Select Goal(s): All, None   Run in Group Mode

Methodology: New_RTL   GuideWare Reference Methodology for New RTL Block development

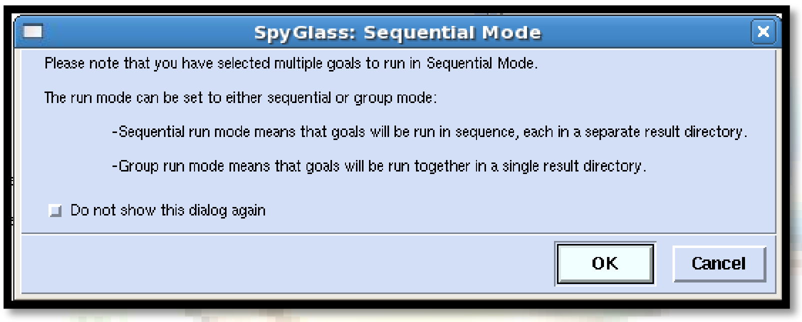
Goal	Setup Status	Run Status	Prereq. Goals
initial_rtl		Selected Goal(s):	
lint			
audit			
clock_reset_integrity		Selected Goal(s):	
power_gated_clock	Setup Recommended	Not Run Yet	
clock_reset_integrity	Setup Done	Not Run Yet	
constraint			
constraint_generation			
power			
voltage_domain			
cdc_prep			
cdc_verif		Selected Goal(s):	
cdc_verif_base	Setup Recommended	Not Run Yet	initial_rtl/clock_reset_integrity/clock_reset
cdc_verif	Setup Recommended	Not Run Yet	initial_rtl/cdc_verif/cdc_verif_base

24. Right-click "Run Selected Goal(s)" to run goals sequentially.


Click on the check box to select or deselect a goal for multiple run.

Currently Selected Goal(s): 3 out of 179

initial_rtl/clock_reset_integrity/clock_reset_integrity
initial_rtl/cdc_verif/cdc_verif_base
initial_rtl/cdc_verif/cdc_verif


SpyGlass: Sequential Mode

Please note that you have selected multiple goals to run in Sequential Mode.

The run mode can be set to either sequential or group mode:

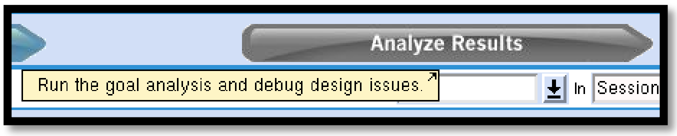
-Sequential run mode means that goals will be run in sequence, each in a separate result directory.

-Group run mode means that goals will be run together in a single result directory.

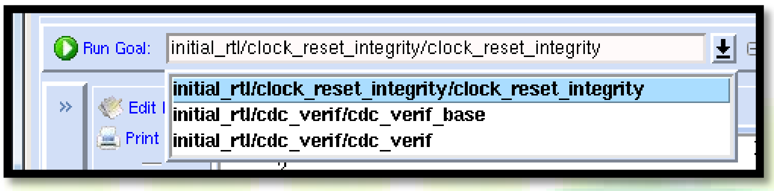
Do not show this dialog again

OK   Cancel

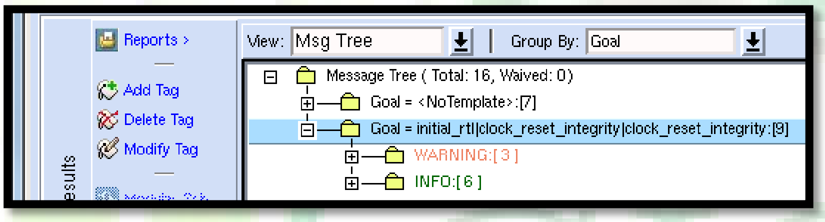
25. Switch to "Analyze Results" section



26. Switch goal to “initial_rtl/clock_reset_intergrity/clock_reset_integrity” and then check warnings and errors

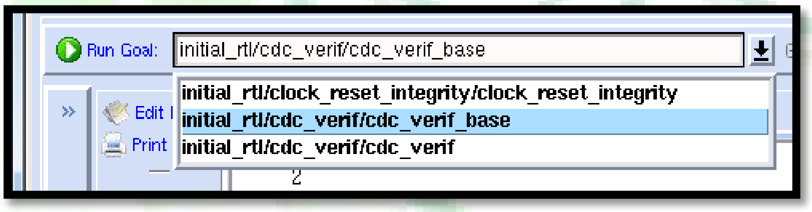


```
initial_rtl/clock_reset_integrity/clock_reset_integrity

initial_rtl/clock_reset_integrity/clock_reset_integrity
initial_rtl/cdc_verif/cdc_verif_base
initial_rtl/cdc_verif/cdc_verif
```

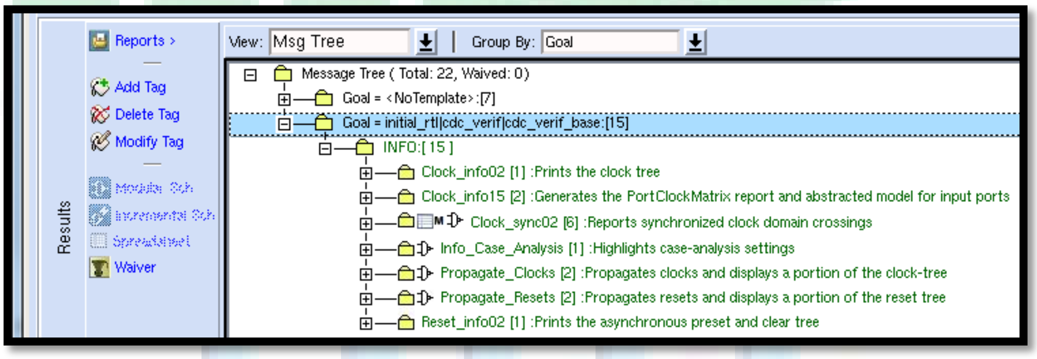


```
Message Tree ( Total: 16, Waived: 0)
  Goal = <NoTemplate>:[7]
  Goal = initial_rtl|clock_reset_integrity|clock_reset_integrity:[9]
    WARNING:[3]
    INFO:[6]
```

27. Switch goal to “initial_rtl/cdc_verif/cdc_verif_base” and then check warnings and errors

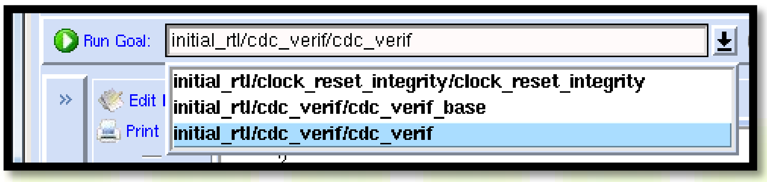


```
initial_rtl/cdc_verif/cdc_verif_base

initial_rtl/clock_reset_integrity/clock_reset_integrity
initial_rtl/cdc_verif/cdc_verif_base
initial_rtl/cdc_verif/cdc_verif
```



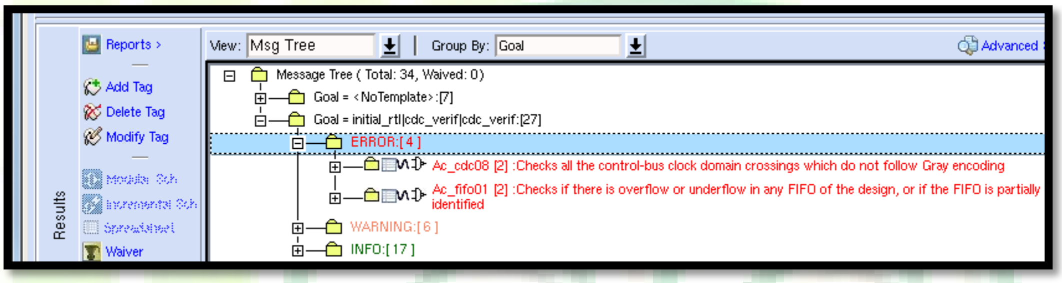
```
Message Tree ( Total: 22, Waived: 0)
  Goal = <NoTemplate>:[7]
  Goal = initial_rtl|cdc_verif|cdc_verif_base:[15]
    INFO:[15]
      Clock_info02 [1] :Prints the clock tree
      Clock_info15 [2] :Generates the PortClockMatrix report and abstracted model for input ports
      Clock_sync02 [6] :Reports synchronized clock domain crossings
      Info_Case_Analysis [1] :Highlights case-analysis settings
      Propagate_Clocks [2] :Propagates clocks and displays a portion of the clock-tree
      Propagate_Resets [2] :Propagates resets and displays a portion of the reset tree
      Reset_info02 [1] :Prints the asynchronous preset and clear tree
```

28. Switch goal to “initial_rtl/cdc_verif/cdc_verif” and then check warnings and errors



initial_rtl/cdc_verif/cdc_verif

initial_rtl/clock_reset_integrity/clock_reset_integrity
initial_rtl/cdc_verif/cdc_verif_base
initial_rtl/cdc_verif/cdc_verif



View: Msg Tree Group By: Goal

```
Message Tree ( Total: 34, Waived: 0)
  Goal = <NoTemplate>:[7]
  Goal = initial_rtl|cdc_verif|cdc_verif:[27]
    ERROR:[ 4 ]
      Ac_cdc08 [2] :Checks all the control-bus clock domain crossings which do not follow Gray encoding
      Ac_fifo01 [2] :Checks if there is overflow or underflow in any FIFO of the design, or if the FIFO is partially identified
    WARNING:[ 6 ]
    INFO:[ 17 ]
```

[Think about it]

Can we waive these errors? If not, how to fix errors?

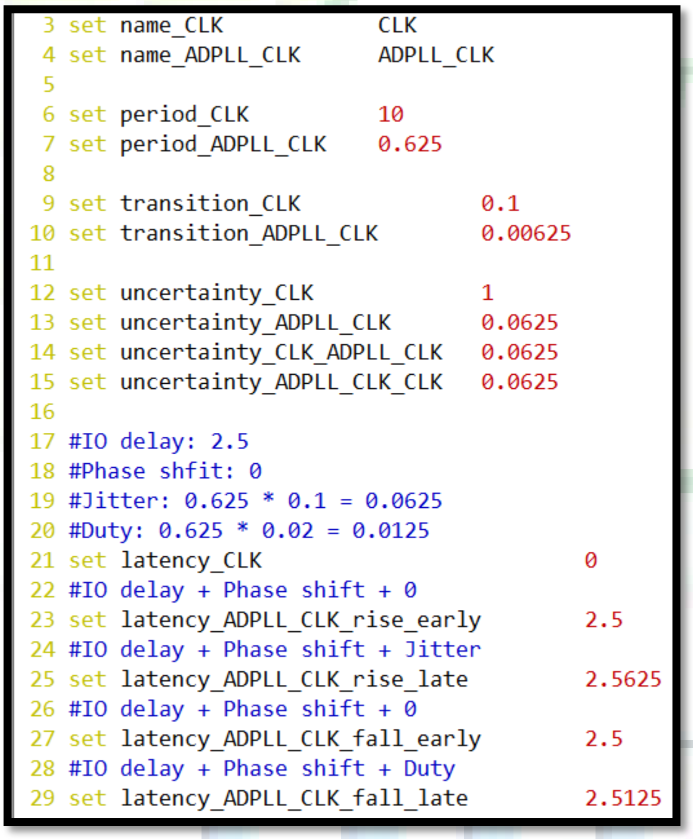
Lab 3 Case study: Synthesis

1. Change directory to lab 3

   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_3

2. Setting SDC variables

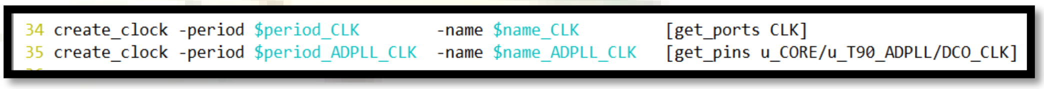
File name	CHIP_dc.tcl
Line number	3~29



```
 3 set name_CLK            CLK
 4 set name_ADPLL_CLK      ADPLL_CLK
 5
 6 set period_CLK          10
 7 set period_ADPLL_CLK    0.625
 8
 9 set transition_CLK              0.1
10 set transition_ADPLL_CLK        0.00625
11
12 set uncertainty_CLK             1
13 set uncertainty_ADPLL_CLK       0.0625
14 set uncertainty_CLK_ADPLL_CLK   0.0625
15 set uncertainty_ADPLL_CLK_CLK   0.0625
16
17 #IO delay: 2.5
18 #Phase shfit: 0
19 #Jitter: 0.625 * 0.1 = 0.0625
20 #Duty: 0.625 * 0.02 = 0.0125
21 set latency_CLK                         0
22 #IO delay + Phase shift + 0
23 set latency_ADPLL_CLK_rise_early        2.5
24 #IO delay + Phase shift + Jitter
25 set latency_ADPLL_CLK_rise_late         2.5625
26 #IO delay + Phase shift + 0
27 set latency_ADPLL_CLK_fall_early        2.5
28 #IO delay + Phase shift + Duty
29 set latency_ADPLL_CLK_fall_late         2.5125
```

3. Setting master clocks

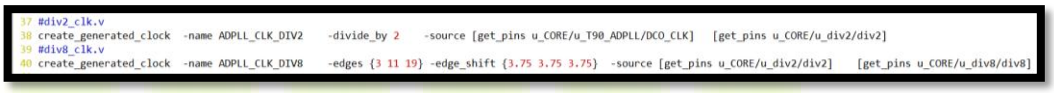
File name	CHIP_dc.tcl
Line number	34 and 35



```
34 create_clock -period $period_CLK        -name $name_CLK        [get_ports CLK]
35 create_clock -period $period_ADPLL_CLK  -name $name_ADPLL_CLK  [get_pins u_CORE/u_T90_ADPLL/DCO_CLK]
```

4. Setting divide clocks

File name	CHIP_dc.tcl
Line number	38 and 40



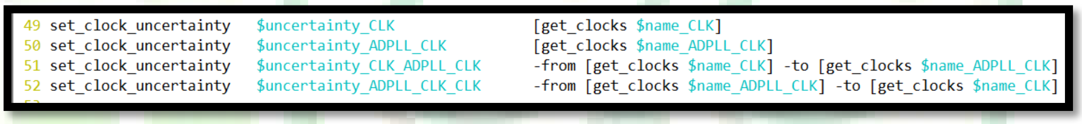
```
37 #div2_clk.v
38 create_generated_clock  -name ADPLL_CLK_DIV2    -divide_by 2    -source [get_pins u_CORE/u_T90_ADPLL/DCO_CLK]    [get_pins u_CORE/u_div2/div2]
39 #div8_clk.v
40 create_generated_clock  -name ADPLL_CLK_DIV8    -edges {3 11 19} -edge_shift {3.75 3.75 3.75}  -source [get_pins u_CORE/u_div2/div2]    [get_pins u_CORE/u_div8/div8]
```

[Think about it]

Why "ADPLL_CLK_DIV8" use "edges" and "edge_shift"?

5. Setting clock uncertainty

File name	CHIP_dc.tcl
Line number	49 ~ 52



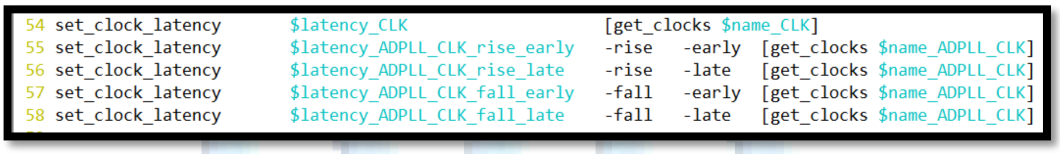
```
49 set_clock_uncertainty   $uncertainty_CLK              [get_clocks $name_CLK]
50 set_clock_uncertainty   $uncertainty_ADPLL_CLK        [get_clocks $name_ADPLL_CLK]
51 set_clock_uncertainty   $uncertainty_CLK_ADPLL_CLK    -from [get_clocks $name_CLK] -to [get_clocks $name_ADPLL_CLK]
52 set_clock_uncertainty   $uncertainty_ADPLL_CLK_CLK    -from [get_clocks $name_ADPLL_CLK] -to [get_clocks $name_CLK]
```

[Think about it]

What is "set_clock_uncertainty –from clka –to clkb] means?

6. Setting clock latency

File name	CHIP_dc.tcl
Line number	54 ~ 58



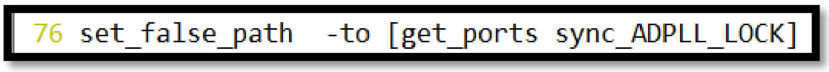
```
54 set_clock_latency      $latency_CLK                     [get_clocks $name_CLK]
55 set_clock_latency      $latency_ADPLL_CLK_rise_early    -rise   -early  [get_clocks $name_ADPLL_CLK]
56 set_clock_latency      $latency_ADPLL_CLK_rise_late     -rise   -late   [get_clocks $name_ADPLL_CLK]
57 set_clock_latency      $latency_ADPLL_CLK_fall_early    -fall   -early  [get_clocks $name_ADPLL_CLK]
58 set_clock_latency      $latency_ADPLL_CLK_fall_late     -fall   -late   [get_clocks $name_ADPLL_CLK]
```

[Think about it]

Why we need "rise/fall" and "early/late" options?

7. Setting false path

File name	CHIP_dc.tcl
Line number	76



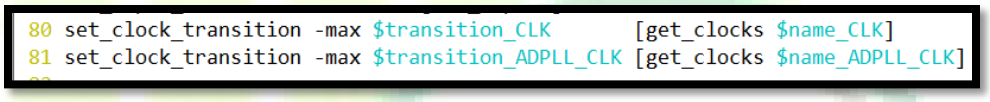
```
76 set_false_path  -to [get_ports sync_ADPLL_LOCK]
```

[Think about it]

Why we can set “sync_ADPLL_LOCK” to false path?

8. Setting clock transitions

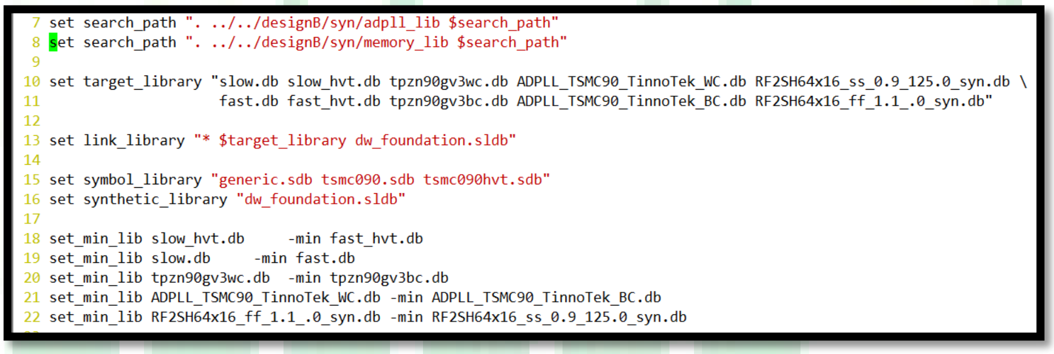
File name	CHIP_dc.tcl
Line number	80 and 81



```
80 set_clock_transition -max $transition_CLK       [get_clocks $name_CLK]
81 set_clock_transition -max $transition_ADPLL_CLK [get_clocks $name_ADPLL_CLK]
```

9. Setting ADPLL IP library

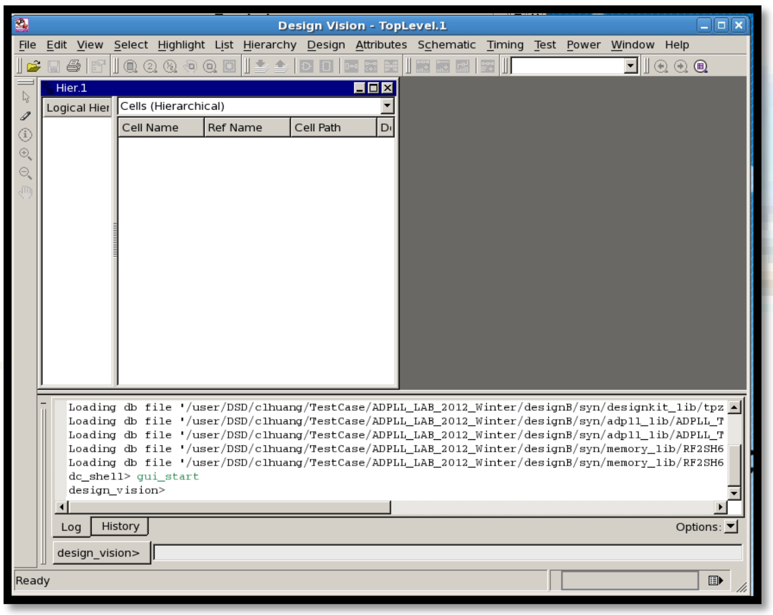
File name	.synopsys_dc_.setup
Line number	7, 10, 11 and 21



```
 7 set search_path ". ../../designB/syn/adpll_lib $search_path"
 8 set search_path ". ../../designB/syn/memory_lib $search_path"
 9
10 set target_library "slow.db slow_hvt.db tpzn90gv3wc.db ADPLL_TSMC90_TinnoTek_WC.db RF2SH64x16_ss_0.9_125.0_syn.db \
11                     fast.db fast_hvt.db tpzn90gv3bc.db ADPLL_TSMC90_TinnoTek_BC.db RF2SH64x16_ff_1.1_.0_syn.db"
12
13 set link_library "* $target_library dw_foundation.sldb"
14
15 set symbol_library "generic.sdb tsmc090.sdb tsmc090hvt.sdb"
16 set synthetic_library "dw_foundation.sldb"
17
18 set_min_lib slow_hvt.db     -min fast_hvt.db
19 set_min_lib slow.db     -min fast.db
20 set_min_lib tpzn90gv3wc.db  -min tpzn90gv3bc.db
21 set_min_lib ADPLL_TSMC90_TinnoTek_WC.db -min ADPLL_TSMC90_TinnoTek_BC.db
22 set_min_lib RF2SH64x16_ff_1.1_.0_syn.db -min RF2SH64x16_ss_0.9_125.0_syn.db
```

10. Open the design compiler in GUI mode

Unix% dv

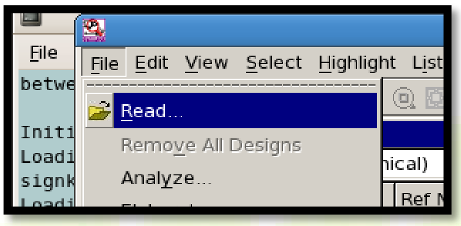


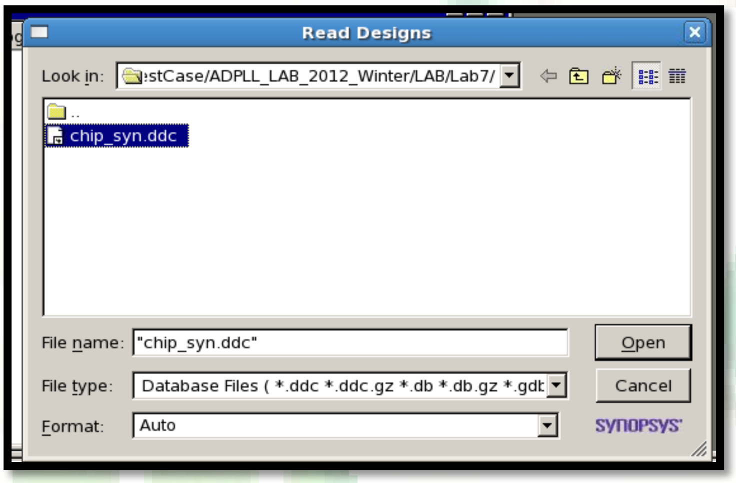
Design Vision - TopLevel.1
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power Window Help
Hier.1
Logical Hier
Cells (Hierarchical)
Cell Name
Ref Name
Cell Path
Log
History
Options:
design_vision>
Ready

```
Loading db file '/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/syn/designkit_lib/tpz
Loading db file '/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/syn/adpll_lib/ADPLL_T
Loading db file '/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/syn/adpll_lib/ADPLL_T
Loading db file '/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/syn/memory_lib/RF2SH6
Loading db file '/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/syn/memory_lib/RF2SH6
dc_shell> gui_start
design_vision>
```

11. Reading design

File => Read... => chip_syn.ddc





Look in: stCase/ADPLL_LAB_2012_Winter/LAB/Lab7/

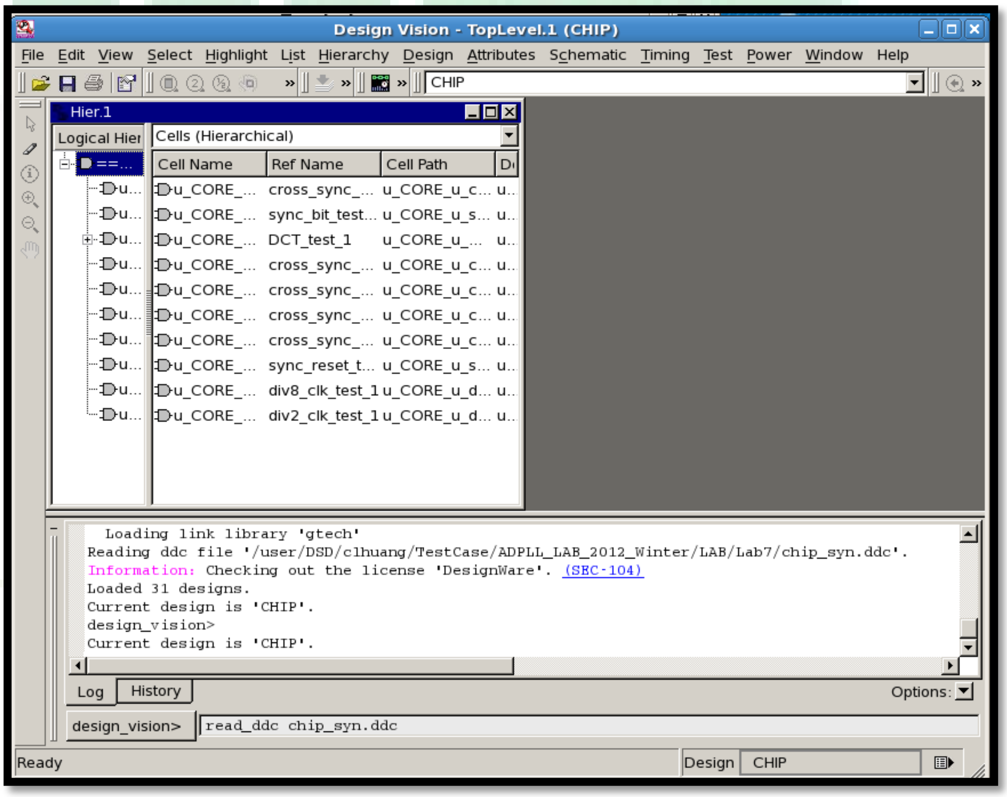
chip_syn.ddc

File name: "chip_syn.ddc"

File type: Database Files (*.ddc *.ddc.gz *.db *.db.gz *.gdt

Format: Auto

12. And then you can review the design and analysis timing.


Design Vision - TopLevel.1 (CHIP)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power Window Help
CHIP
Hier.1
Logical Hier
Cells (Hierarchical)
Log
History
Options:
Ready
Design CHIP

Cell Name	Ref Name	Cell Path	D
u_CORE_...	cross_sync_...	u_CORE_u_c...	u..
u_CORE_...	sync_bit_test...	u_CORE_u_s...	u..
u_CORE_...	DCT_test_1	u_CORE_u_...	u..
u_CORE_...	cross_sync_...	u_CORE_u_c...	u..
u_CORE_...	cross_sync_...	u_CORE_u_c...	u..
u_CORE_...	cross_sync_...	u_CORE_u_c...	u..
u_CORE_...	cross_sync_...	u_CORE_u_c...	u..
u_CORE_...	sync_reset_t...	u_CORE_u_s...	u..
u_CORE_...	div8_clk_test_1	u_CORE_u_d...	u..
u_CORE_...	div2_clk_test_1	u_CORE_u_d...	u..

```
  Loading link library 'gtech'
Reading ddc file '/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/LAB/Lab7/chip_syn.ddc'.
Information: Checking out the license 'DesignWare'. (SEC-104)
Loaded 31 designs.
Current design is 'CHIP'.
design_vision>
Current design is 'CHIP'.
```

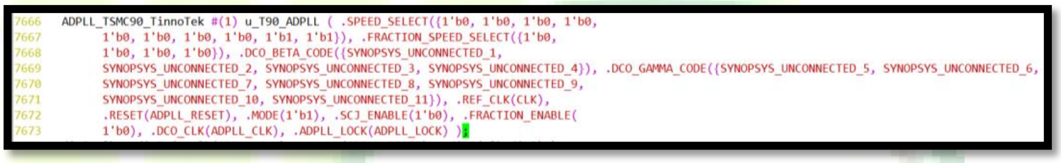
```
design_vision> read_ddc chip_syn.ddc
```

13. Close design compiler and open the synthesized Verilog netlist.

File name	chip_syn.v

Because the post-synthesis simulation use pseudo-random jitter mode, we need to add a parameter in Verilog netlist file.

Line number	7666



```
7666   ADPLL_TSMC90_TinnoTek #(1) u_T90_ADPLL ( .SPEED_SELECT({1'b0, 1'b0, 1'b0, 1'b0,
7667         1'b0, 1'b0, 1'b0, 1'b0, 1'b1, 1'b1}), .FRACTION_SPEED_SELECT({1'b0,
7668         1'b0, 1'b0, 1'b0}), .DCO_BETA_CODE({SYNOPSYS_UNCONNECTED_1,
7669         SYNOPSYS_UNCONNECTED_2, SYNOPSYS_UNCONNECTED_3, SYNOPSYS_UNCONNECTED_4}), .DCO_GAMMA_CODE({SYNOPSYS_UNCONNECTED_5, SYNOPSYS_UNCONNECTED_6,
7670         SYNOPSYS_UNCONNECTED_7, SYNOPSYS_UNCONNECTED_8, SYNOPSYS_UNCONNECTED_9,
7671         SYNOPSYS_UNCONNECTED_10, SYNOPSYS_UNCONNECTED_11}), .REF_CLK(CLK),
7672         .RESET(ADPLL_RESET), .MODE(1'b1), .SCJ_ENABLE(1'b0), .FRACTION_ENABLE(
7673         1'b0), .DCO_CLK(ADPLL_CLK), .ADPLL_LOCK(ADPLL_LOCK) );
```

[Think about it]

Open the VCD waveform file, can you tell difference between ideal jitter mode and pseudo-random jitter mode?

Lab 4A Case study: Layout (SOC Encounter)

1. Change directory to lab 4A

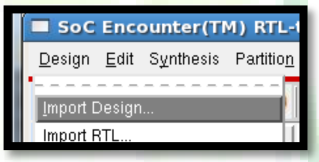
   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_4A

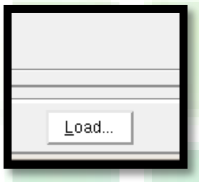
2. Open the SOC Encounter

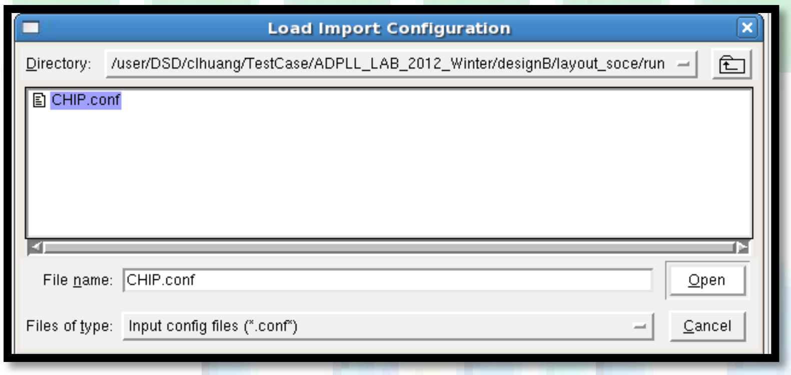
   Unix% encounter

3. Import Design by loading configure file

Configure file name	CHIP.conf







Directory: /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/layout_soce/run

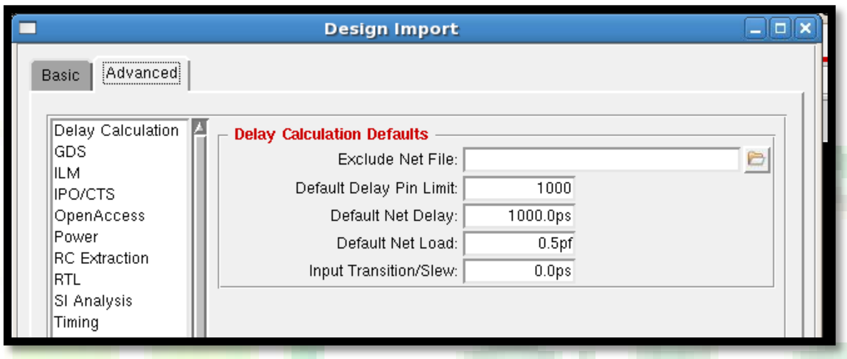
CHIP.conf

File name: CHIP.conf

Files of type: Input config files (*.conf*)

4. Because we provides a independent power domain for ADPLL IP, we needs extra configuration.

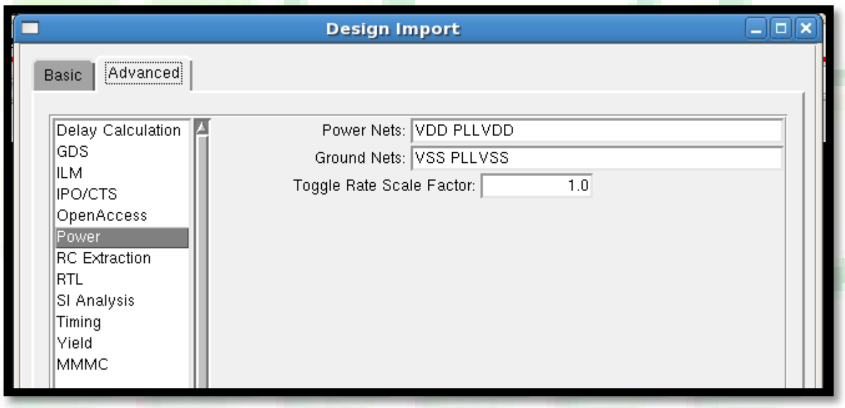
   In the “Design import” window, switch tab to “Advanced”.



Delay Calculation Defaults

Exclude Net File:
Default Delay Pin Limit: 1000
Default Net Delay: 1000.0ps
Default Net Load: 0.5pf
Input Transition/Slew: 0.0ps

   At “Power” section, add another power/ground name for ADPLL IP

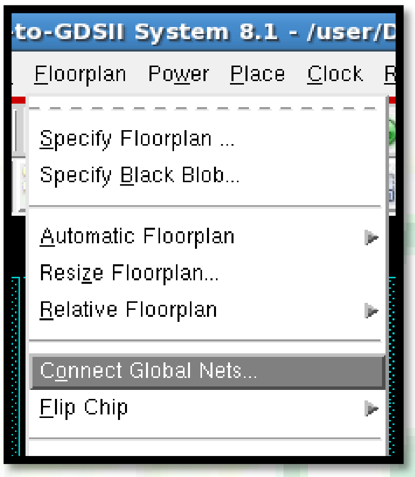


Power Nets: VDD PLLVDD
Ground Nets: VSS PLLVSS
Toggle Rate Scale Factor: 1.0

5. Click “OK” to perform design import

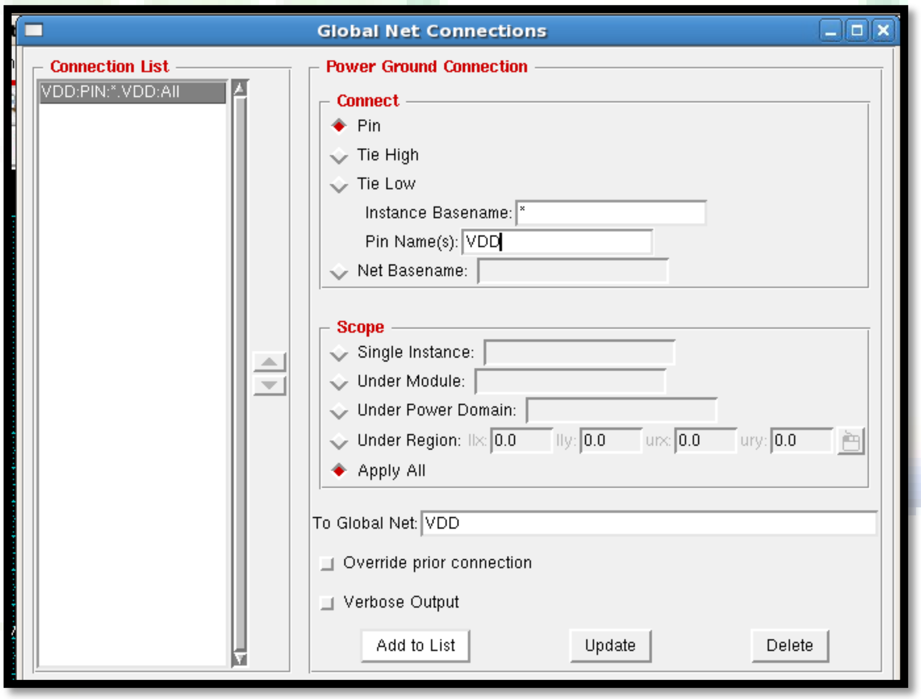
6. Global net connection.

Floorplan => Connect Global Nets…



First, connect global power net to "VDD"

Pin name	VDD
Scope	Apply All
To Global Net	VDD


Global Net Connections

Connection List

VDD:PIN:*.VDD:All

Power Ground Connection

Connect
- [✓ Pin]
- Tie High
- Tie Low

Instance Basename: *

Pin Name(s): VDD

Net Basename:

Scope
- Single Instance:
- Under Module:
- Under Power Domain:
- Under Region: llx: 0.0 lly: 0.0 urx: 0.0 ury: 0.0
- [✓ Apply All]

To Global Net: VDD

Override prior connection

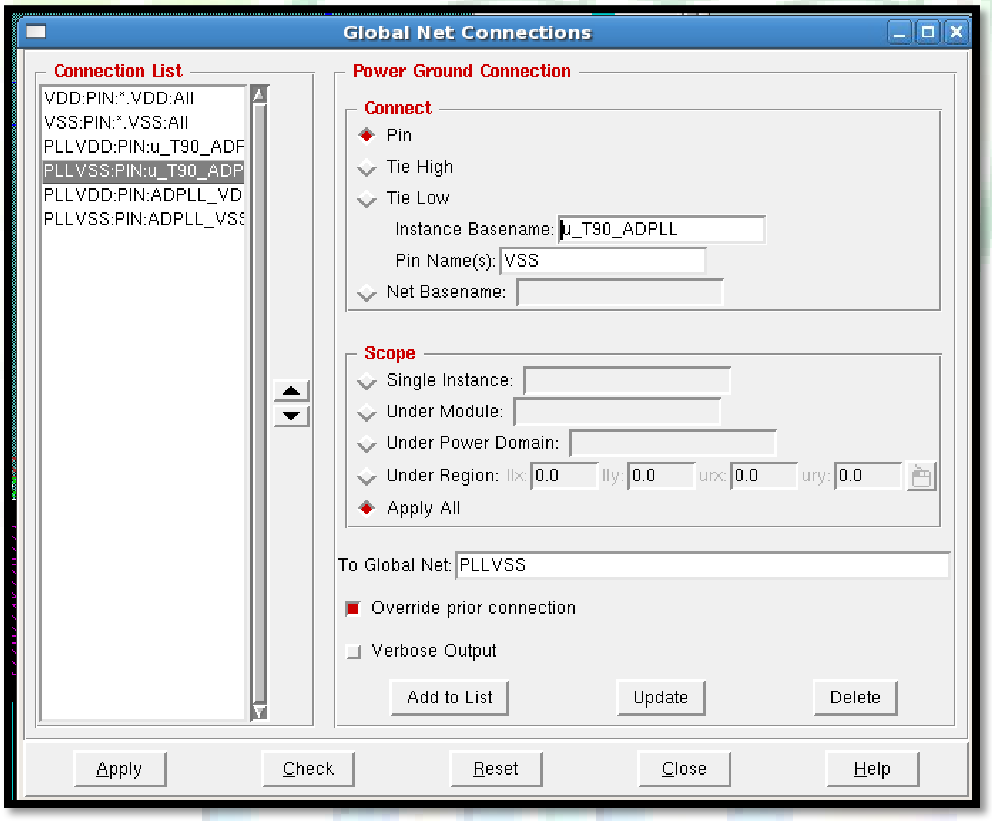
Verbose Output

Connect global ground net to “VSS”

Pin name	VSS
Scope	Apply All
To Global Net	VSS

Connect ADPLL IP power net to “PLLVDD”

Instance Basename	u_T90_ADPLL
Pin Name(s)	VDD
Scope	Apply All
To Global Net	PLLVDD
Override prior connection	Check



```
VDD:PIN:*.VDD:All
VSS:PIN:*.VSS:All
PLLVDD:PIN:u_T90_ADF
PLLVSS:PIN:u_T90_ADP
PLLVDD:PIN:ADPLL_VD
PLLVSS:PIN:ADPLL_VSS
```

Connect: [✓ Pin] Tie High Tie Low
Instance Basename: u_T90_ADPLL
Pin Name(s): VSS
Net Basename:

Scope: Single Instance: Under Module: Under Power Domain: Under Region: llx: 0.0 lly: 0.0 urx: 0.0 ury: 0.0 [✓ Apply All]

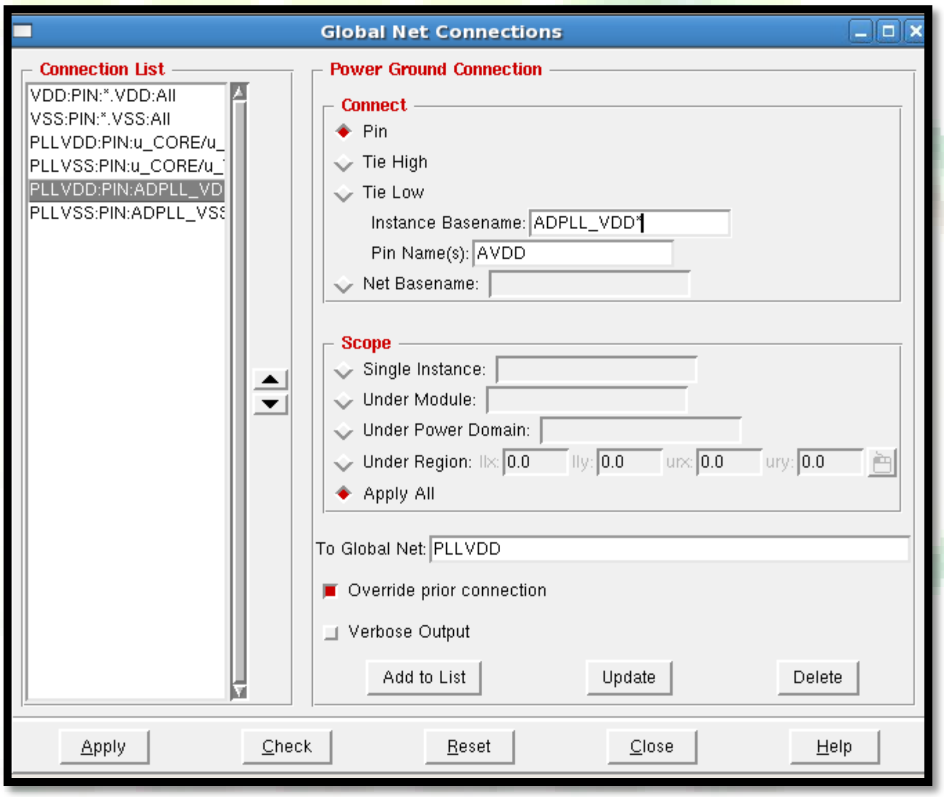
To Global Net: PLLVSS
[✓ Override prior connection]
Verbose Output

Connect ADPLL IP ground net to “PLLVSS”

Instance Basename	u_T90_ADPLL
Pin Name(s)	VSS
Scope	Apply All
To Global Net	PLLVSS
Override prior connection	Check

Connect ADPLL power pad to “PLLVDD”

Instance Basename	ADPLL_VDD*
Pin Name(s)	AVDD
Scope	Apply All
To Global Net	PLLVDD
Override prior connection	Check


Global Net Connections

Connection List
VDD:PIN:*.VDD:All
VSS:PIN:*.VSS:All
PLLVDD:PIN:u_CORE/u_
PLLVSS:PIN:u_CORE/u_
PLLVDD:PIN:ADPLL_VD
PLLVSS:PIN:ADPLL_VSS

Power Ground Connection
Connect
◆ Pin
◇ Tie High
◇ Tie Low
Instance Basename: ADPLL_VDD*
Pin Name(s): AVDD
◇ Net Basename:

Scope
◇ Single Instance:
◇ Under Module:
◇ Under Power Domain:
◇ Under Region: llx: 0.0 lly: 0.0 urx: 0.0 ury: 0.0
◆ Apply All

To Global Net: PLLVDD
■ Override prior connection
☐ Verbose Output

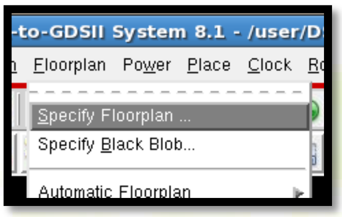
Connect ADPLL power pad to “PLLVSS”

Instance Basename	ADPLL_VSS*
Pin Name(s)	AVSS
Scope	Apply All
To Global Net	PLLVSS
Override prior connection	Check

Apply and check global net

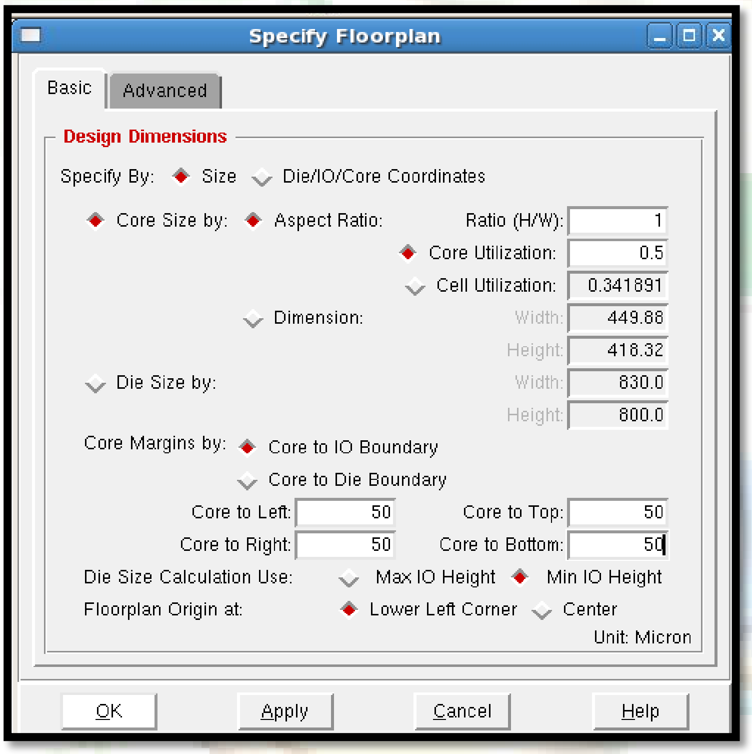
7. Floorplan

Floorplan => Specify Floorplan…



Setting design dimensions

Ratio (H/W)	1
Core Utilization	1
Core to Left	50
Core to Top	50
Core to Right	50
Core to Bottom	50


Specify Floorplan

Basic | Advanced

Design Dimensions

Specify By: [✓ Size] Die/IO/Core Coordinates

[✓ Core Size by:] [✓ Aspect Ratio:] Ratio (H/W): 1

[✓ Core Utilization:] 0.5

Cell Utilization: 0.341891

Dimension: Width: 449.88 Height: 418.32

Die Size by: Width: 830.0 Height: 800.0

Core Margins by: [✓ Core to IO Boundary] Core to Die Boundary

Core to Left: 50 Core to Top: 50

Core to Right: 50 Core to Bottom: 50

Die Size Calculation Use: Max IO Height [✓ Min IO Height]

Floorplan Origin at: [✓ Lower Left Corner] Center

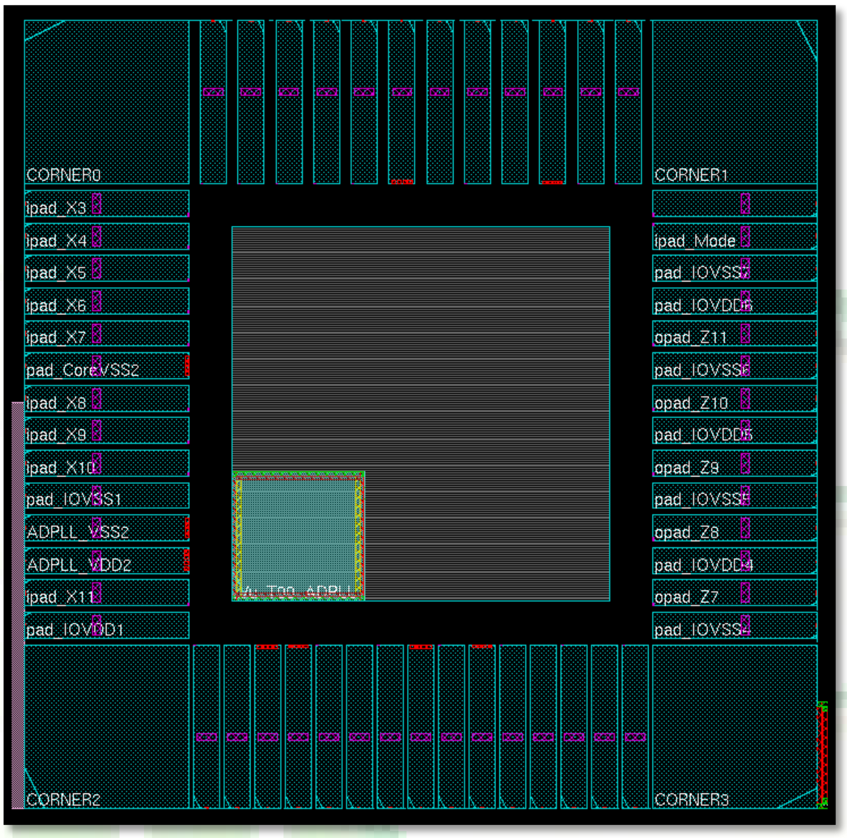
Unit: Micron

OK Apply Cancel Help

[Notice]

Do NOT use this floorplan configuration for real design!

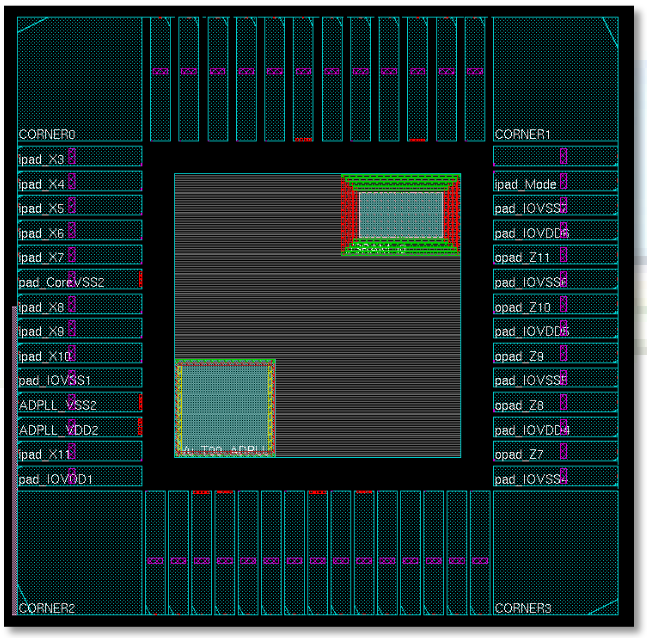
8. Move ADPLL IP into floorplan as shown below



[Think about it]

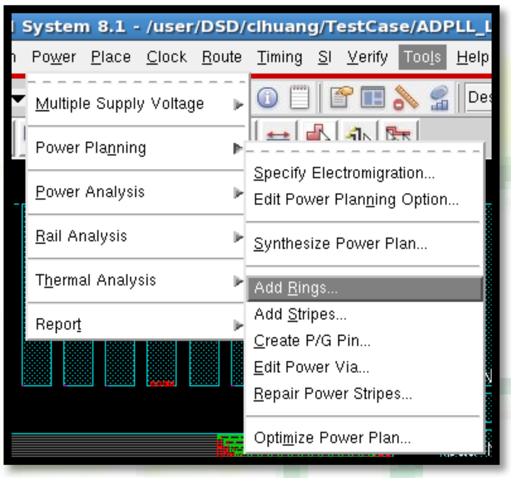
Why we move ADPLL IP to the corner?

9. Place other IPs



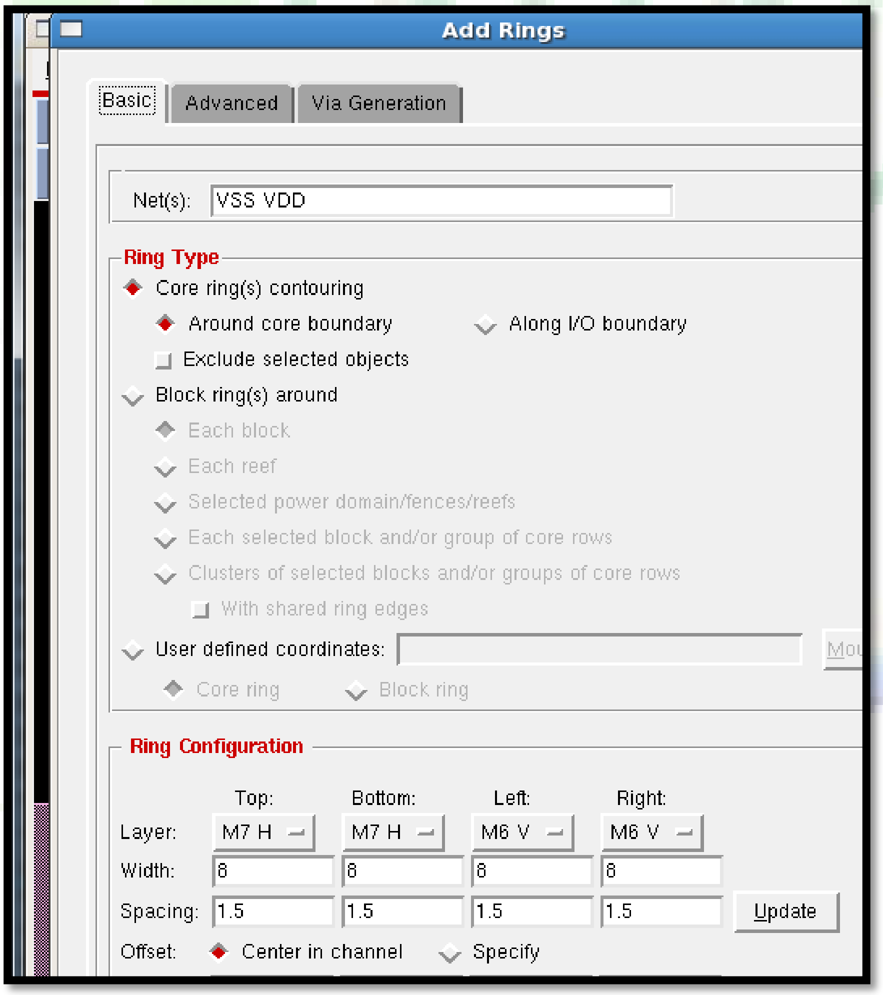
10. Create power ring.

Power => Power Planning => Add Rings…



Basic power ring configuration

Net(s)	VSS VDD
Top Layer	M7
Top Width	8
Top Spacing	1.5
Bottom Layer	M7
Bottom Width	8
Bottom Spacing	1.5
Left Layer	M6
Left Width	8
Left Spacing	1.5
Right Layer	M6
Right Width	8
Right Spacing	1.5
Offset	Center in channel


Add Rings

Basic | Advanced | Via Generation

Net(s): VSS VDD

Ring Type
- [✓ Core ring(s) contouring]
  - [✓ Around core boundary] Along I/O boundary
  - Exclude selected objects
- Block ring(s) around
  - Each block
  - Each reef
  - Selected power domain/fences/reefs
  - Each selected block and/or group of core rows
  - Clusters of selected blocks and/or groups of core rows
    - With shared ring edges
- User defined coordinates:
  - Core ring Block ring

Ring Configuration

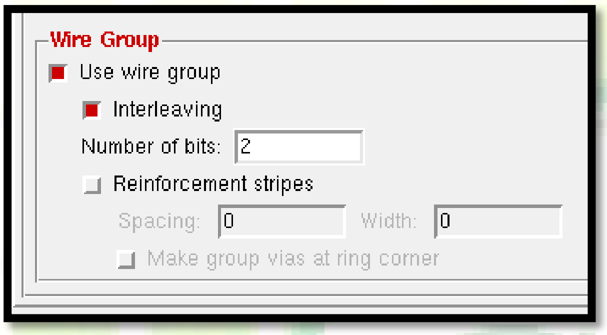
	Top:	Bottom:	Left:	Right:
Layer:	M7 H	M7 H	M6 V	M6 V
Width:	8	8	8	8
Spacing:	1.5	1.5	1.5	1.5

Update

Offset: [✓ Center in channel] Specify

Advanced power ring configuration

Use wire group	Check
Interleaving	Check
Number of but	2


Wire Group

- [✓] Use wire group
- [✓] Interleaving

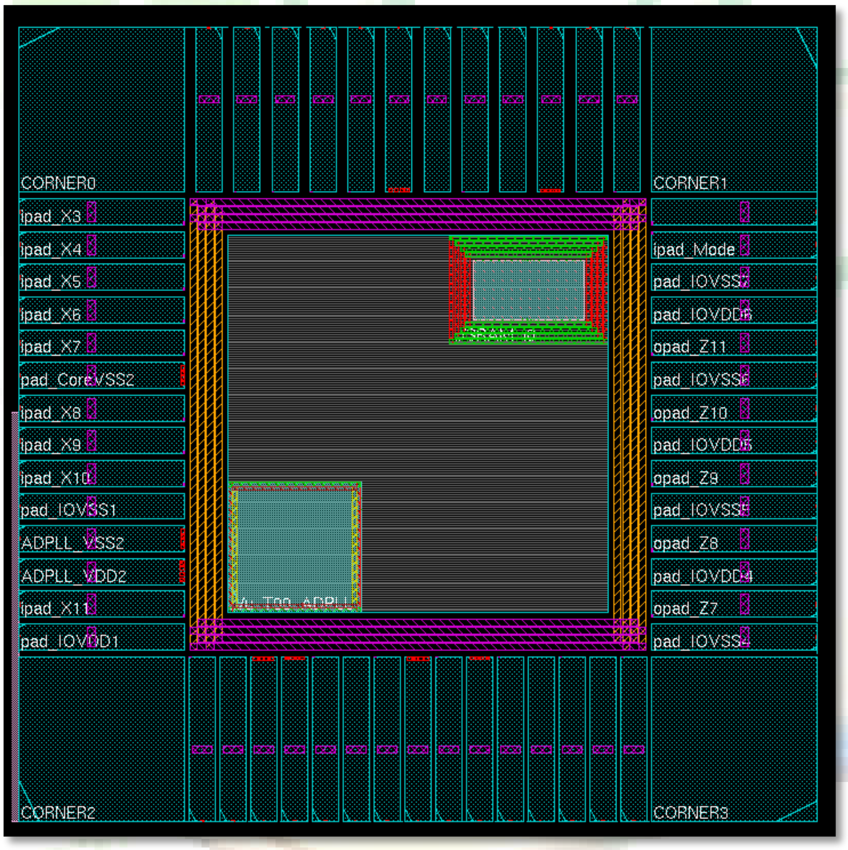
Number of bits: 2

- [ ] Reinforcement stripes

Spacing: 0 Width: 0

- [ ] Make group vias at ring corner

Click "OK" to create power ring.

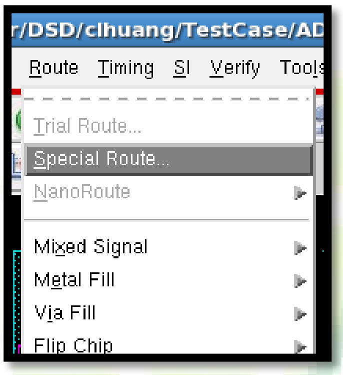


[Think about it]

Why we don't need to create power ring for "PLLVDD/PLLVSS"?

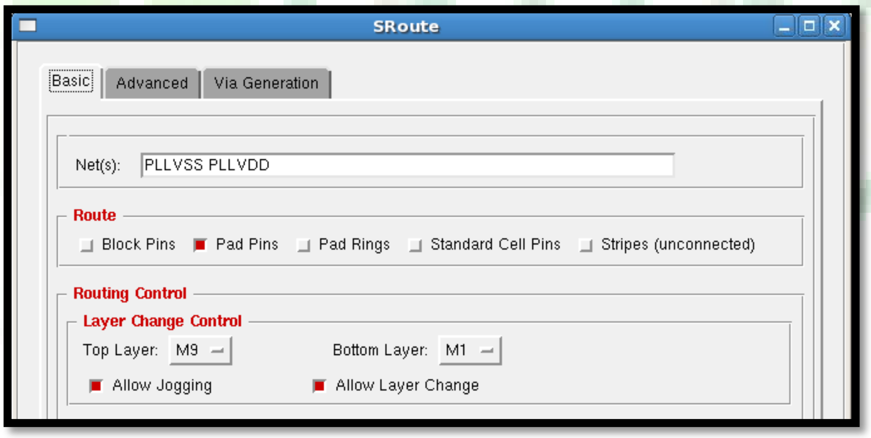
11. Connect ADPLL IP to their own power/ground pads

Route => Special Route…



At Basic SRoute configuration tab

Net(s):	PLLVSS PLLVDD
Route	Pad Pins



Net(s): PLLVSS PLLVDD

Route

Block Pins [✓ Pad Pins] Pad Rings Standard Cell Pins Stripes (unconnected)

Routing Control

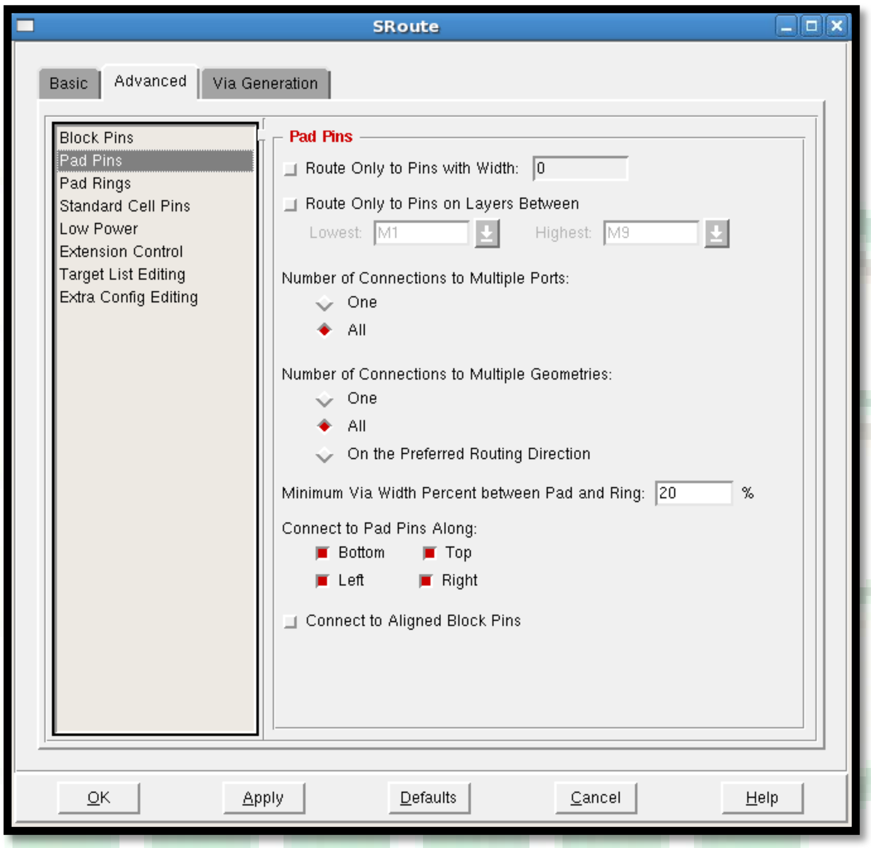
Layer Change Control

Top Layer: M9 Bottom Layer: M1

[✓ Allow Jogging] [✓ Allow Layer Change]

At Advanced SRoute configuration tab and go to "Pad Pins" section

Number of Connections to Multiple Geometries	All
Others	Default



Pad Pins

- [ ] Route Only to Pins with Width: 0
- [ ] Route Only to Pins on Layers Between
  Lowest: M1 Highest: M9

Number of Connections to Multiple Ports: One [✓ All]

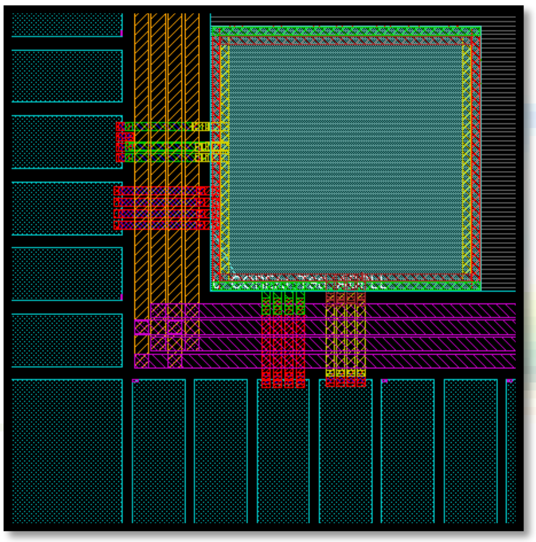
Number of Connections to Multiple Geometries: One [✓ All] On the Preferred Routing Direction

Minimum Via Width Percent between Pad and Ring: 20 %

Connect to Pad Pins Along: [✓ Bottom] [✓ Top] [✓ Left] [✓ Right]

- [ ] Connect to Aligned Block Pins

Click "Apply" to perform SRoute



12. Connect power ring to power/ground pads

At Basic SRoute configuration tab

Net(s):	VDD VSS
Route	Pad Pins

13. There is nothing special after ADPLL IP placement and power planning.

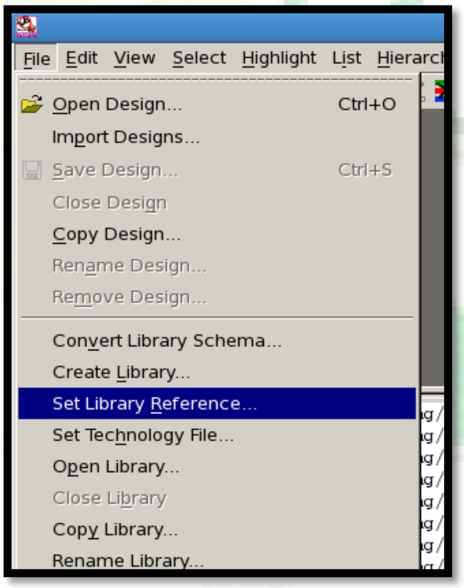
Lab 4B Case study: Layout (IC Compiler)

1. Change directory to lab 4B
   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_4B

2. Open the SpyGlass in GUI mode
   Unix% ic_shell -gui

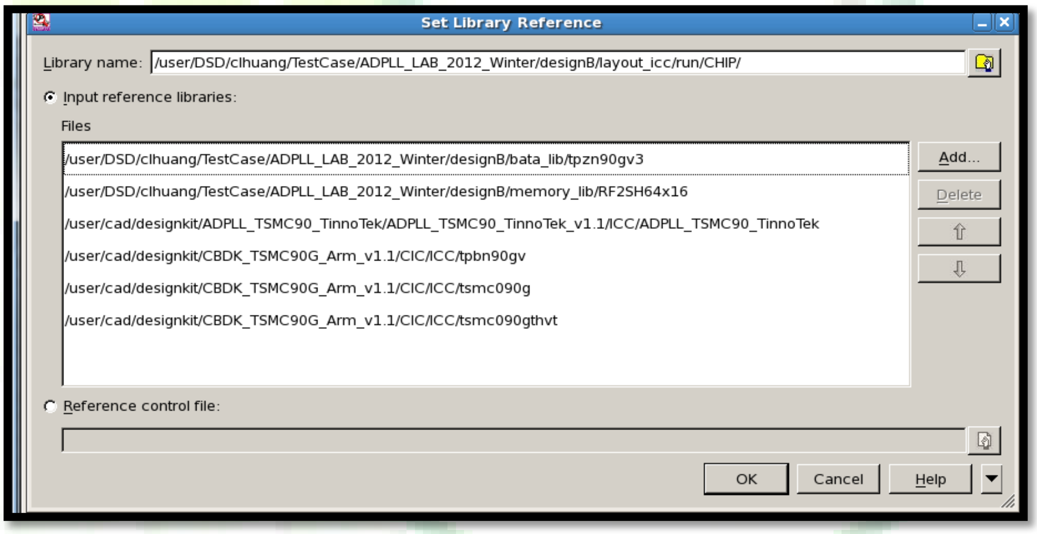
3. Setting library reference

   File => Set Technology File…



Setting library reference

Library name	CHIP
Input reference libraries:	../beta_lib/tpzn90gv3 ../memory_lib/RF2SH64x16 ../adpll_lib/ADPLL_TSMC90_TinnoTek ../designkit_lib/tpbn90gv ../designkit_lib/tsmc090g ../designkit_lib/tsmc090gthvt



Library name: /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/layout_icc/run/CHIP/

(● Input reference libraries:

Files

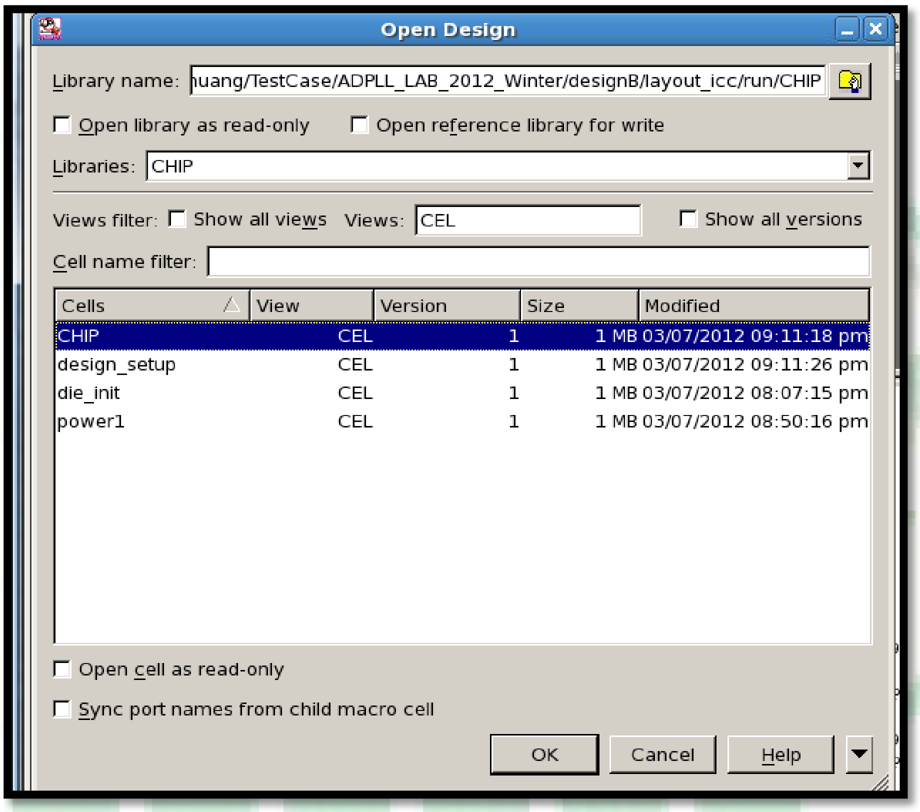
```
/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/bata_lib/tpzn90gv3
/user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/designB/memory_lib/RF2SH64x16
/user/cad/designkit/ADPLL_TSMC90_TinnoTek/ADPLL_TSMC90_TinnoTek_v1.1/ICC/ADPLL_TSMC90_TinnoTek
/user/cad/designkit/CBDK_TSMC90G_Arm_v1.1/CIC/ICC/tpbn90gv
/user/cad/designkit/CBDK_TSMC90G_Arm_v1.1/CIC/ICC/tsmc090g
/user/cad/designkit/CBDK_TSMC90G_Arm_v1.1/CIC/ICC/tsmc090gthvt
```

○ Reference control file:

4. Open design

   File => Open Design

Library name	CHIP
Cells	CHIP



Open Design

Library name: huang/TestCase/ADPLL_LAB_2012_Winter/designB/layout_icc/run/CHIP

☐ Open library as read-only ☐ Open reference library for write

Libraries: CHIP

Views filter: ☐ Show all views Views: CEL ☐ Show all versions

Cell name filter:

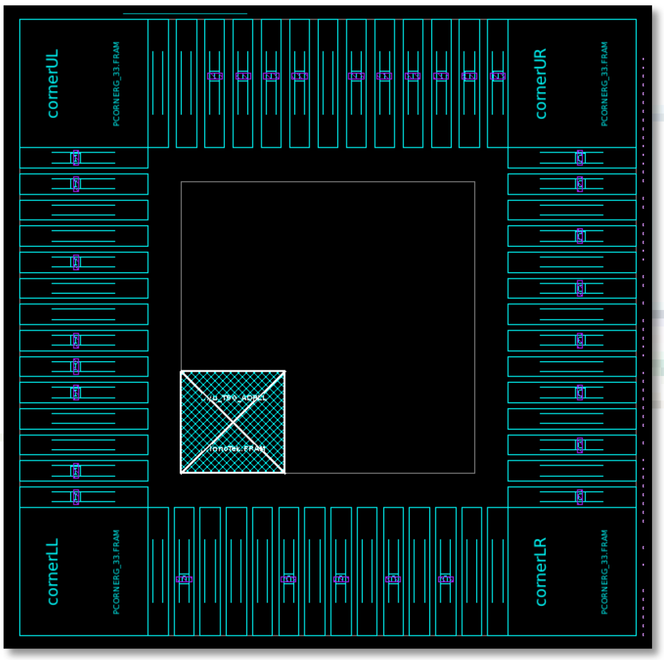
Cells	View	Version	Size	Modified
CHIP	CEL	1	1 MB	03/07/2012 09:11:18 pm
design_setup	CEL	1	1 MB	03/07/2012 09:11:26 pm
die_init	CEL	1	1 MB	03/07/2012 08:07:15 pm
power1	CEL	1	1 MB	03/07/2012 08:50:16 pm

☐ Open cell as read-only

☐ Sync port names from child macro cell

OK Cancel Help

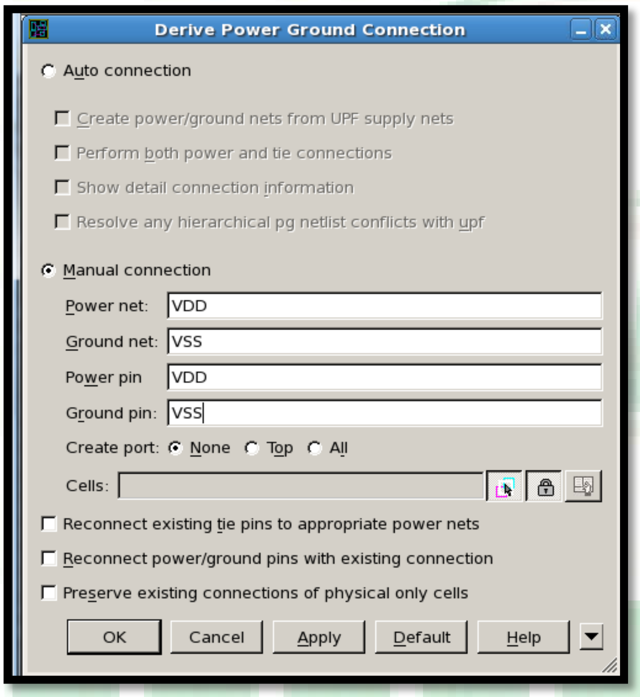
5. Move ADPLL IP to corner as shown below, and then fixed placement



6. Perform Global PG net connection (VDD/VSS)

   Preroute => Derive PG Connection…

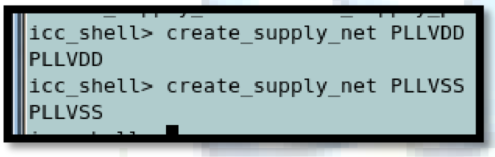
Manual connect	Check
Power net	VDD
Ground net	VSS
Power pin	VDD
Ground pin	VSS


Derive Power Ground Connection
Auto connection
Create power/ground nets from UPF supply nets
Perform both power and tie connections
Show detail connection information
Resolve any hierarchical pg netlist conflicts with upf
Manual connection
Power net: VDD
Ground net: VSS
Power pin VDD
Ground pin: VSS
Create port: [✓ None] Top All
Cells:
Reconnect existing tie pins to appropriate power nets
Reconnect power/ground pins with existing connection
Preserve existing connections of physical only cells
OK
Cancel
Apply
Default
Help

7. Create another supply net for ADPLL IP

   icc_shell> create_supply_net PLLVDD

   icc_shell> create_supply_net PLLVSS

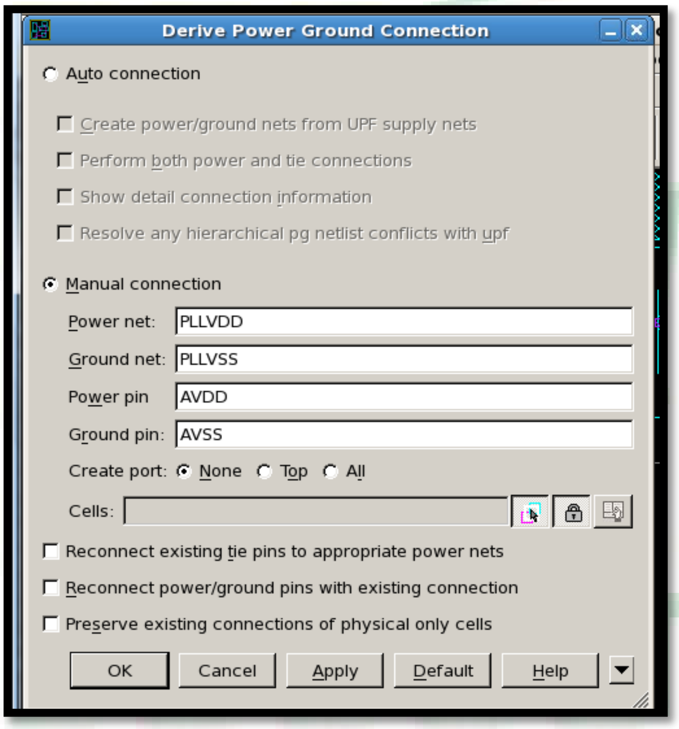


```
icc_shell> create_supply_net PLLVDD
PLLVDD
icc_shell> create_supply_net PLLVSS
PLLVSS
```

8. Perform "PVDD1ANA_33/PVSS1ANA_33" PG net connect (PLLVDD/PLLVSS)

   Preroute => Derive PG Connection…

Manual connect	Check
Power net	PLLVDD
Ground net	PLLVSS
Power pin	AVDD
Ground pin	AVSS



Derive Power Ground Connection

( ) Auto connection

- [ ] Create power/ground nets from UPF supply nets
- [ ] Perform both power and tie connections
- [ ] Show detail connection information
- [ ] Resolve any hierarchical pg netlist conflicts with upf

(●) Manual connection

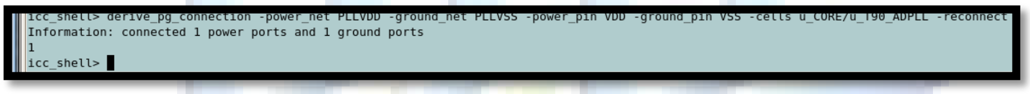
Power net:	PLLVDD
Ground net:	PLLVSS
Power pin	AVDD
Ground pin:	AVSS
Create port:	[✓ None] Top All
Cells:	

- [ ] Reconnect existing tie pins to appropriate power nets
- [ ] Reconnect power/ground pins with existing connection
- [ ] Preserve existing connections of physical only cells

9. Perform ADPLL IP PG net connect (PLLVDD/PLLVSS)

   *icc_shell> derive_pg_connection -power_net PLLVDD -ground_net PLLVSS *

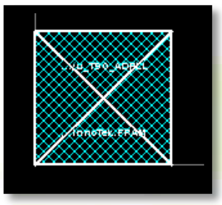
   -power_pin VDD -ground_pin VSS -cells u_CORE/u_T90_ADPLL –reconnect



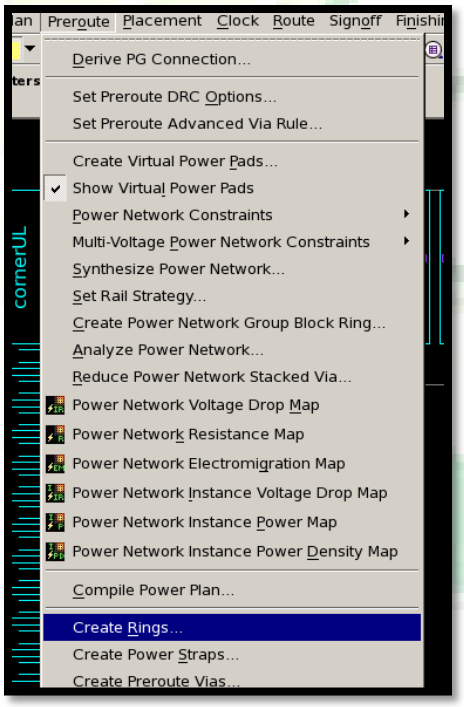
```
icc_shell> derive_pg_connection -power_net PLLVDD -ground_net PLLVSS -power_pin VDD -ground_pin VSS -cells u_CORE/u_T90_ADPLL -reconnect
Information: connected 1 power ports and 1 ground ports
1
icc_shell>
```

10. Create a block ring for ADPLL IP

Highlighted the ADPLL IP.

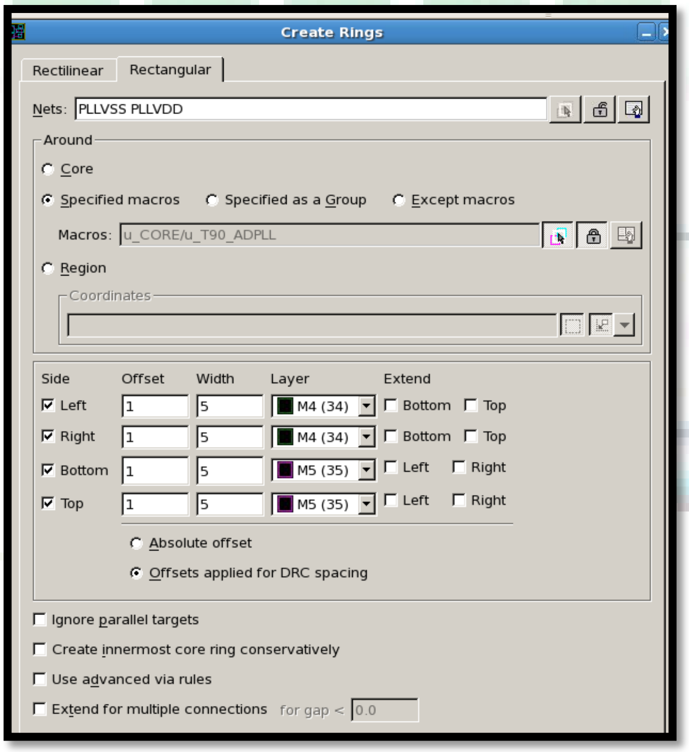


Preroute => Create Rings


Preroute
Placement
Clock
Route
Signoff
Derive PG Connection...
Set Preroute DRC Options...
Set Preroute Advanced Via Rule...
Create Virtual Power Pads...
Show Virtual Power Pads
Power Network Constraints
Multi-Voltage Power Network Constraints
Synthesize Power Network...
Set Rail Strategy...
Create Power Network Group Block Ring...
Analyze Power Network...
Reduce Power Network Stacked Via...
Power Network Voltage Drop Map
Power Network Resistance Map
Power Network Electromigration Map
Power Network Instance Voltage Drop Map
Power Network Instance Power Map
Power Network Instance Power Density Map
Compile Power Plan...
Create Rings...
Create Power Straps...
Create Preroute Vias...
cornerUL

Switch tab to "Rectangular" tab

Nets	PLLVDD PLLVSS
Specified macros	u_CORE/u_T90_ADPLL
Left	Check
Left Offset	1
Left Width	5
Left Layer	M4
Right	Check
Right Offset	1
Right Width	5
Right Layer	M4
Top	Check
Top Offset	1
Top Width	5
Top Layer	M5
Bottom	Check
Bottom Offset	1
Bottom Width	5
Bottom Layer	M5
Others	Default


Create Rings

Rectilinear | Rectangular

Nets: PLLVSS PLLVDD

Around

( ) Core

[✓ Specified macros] ( ) Specified as a Group ( ) Except macros

Macros: u_CORE/u_T90_ADPLL

( ) Region

Coordinates

Side	Offset	Width	Layer	Extend	
☑ Left	1	5	M4 (34)	☐ Bottom	☐ Top
☑ Right	1	5	M4 (34)	☐ Bottom	☐ Top
☑ Bottom	1	5	M5 (35)	☐ Left	☐ Right
☑ Top	1	5	M5 (35)	☐ Left	☐ Right

( ) Absolute offset

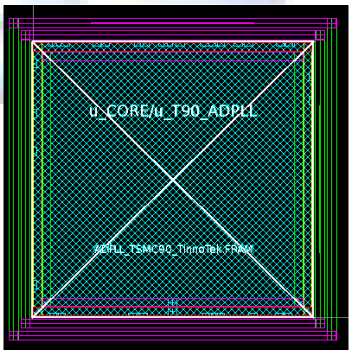
[✓ Offsets applied for DRC spacing]

☐ Ignore parallel targets

☐ Create innermost core ring conservatively

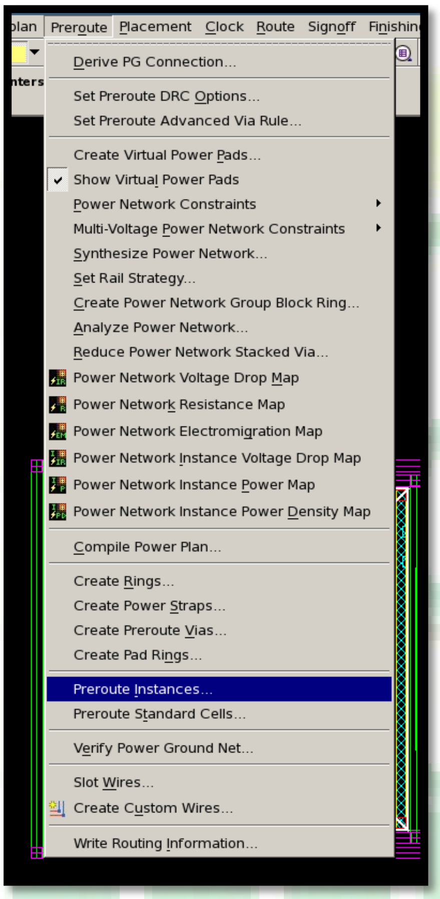
☐ Use advanced via rules

☐ Extend for multiple connections for gap < 0.0



11. Connect “PVDD1ANA/PVSS1ANA” pads to ADPLL block ring

Preroute => Preroute Instances…



plan
Preroute
Placement
Clock
Route
Signoff
Finishin
nters
Derive PG Connection...
Set Preroute DRC Options...
Set Preroute Advanced Via Rule...
Create Virtual Power Pads...
Show Virtual Power Pads
Power Network Constraints
Multi-Voltage Power Network Constraints
Synthesize Power Network...
Set Rail Strategy...
Create Power Network Group Block Ring...
Analyze Power Network...
Reduce Power Network Stacked Via...
Power Network Voltage Drop Map
Power Network Resistance Map
Power Network Electromigration Map
Power Network Instance Voltage Drop Map
Power Network Instance Power Map
Power Network Instance Power Density Map
Compile Power Plan...
Create Rings...
Create Power Straps...
Create Preroute Vias...
Create Pad Rings...
Preroute Instances...
Preroute Standard Cells...
Verify Power Ground Net...
Slot Wires...
Create Custom Wires...
Write Routing Information...

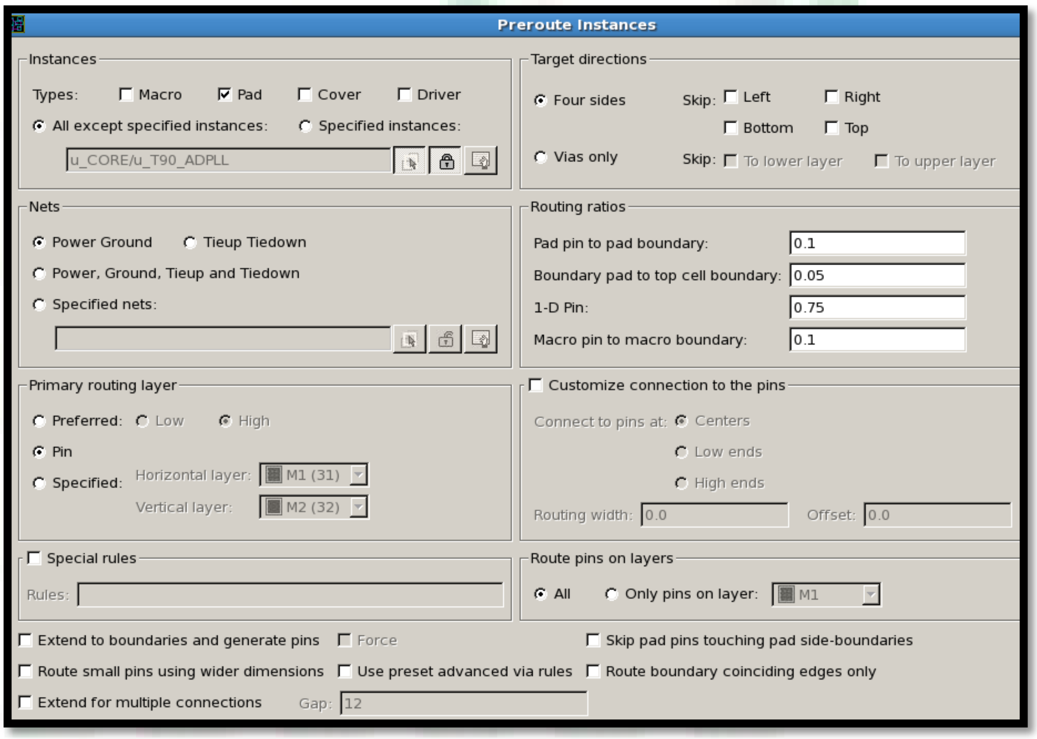
Preroute instances configuration

At “Instances” section

Type	Pad
All except specified instances	check

At “Primary routing layer” section

Pin	Check


Preroute Instances

Instances
Types: ☐ Macro [✓ Pad] ☐ Cover ☐ Driver
[✓ All except specified instances:] ○ Specified instances:
u_CORE/u_T90_ADPLL

Target directions
[✓ Four sides] Skip: ☐ Left ☐ Right ☐ Bottom ☐ Top
○ Vias only Skip: ☐ To lower layer ☐ To upper layer

Nets
[✓ Power Ground] ○ Tieup Tiedown
○ Power, Ground, Tieup and Tiedown
○ Specified nets:

Routing ratios

Pad pin to pad boundary:	0.1
Boundary pad to top cell boundary:	0.05
1-D Pin:	0.75
Macro pin to macro boundary:	0.1

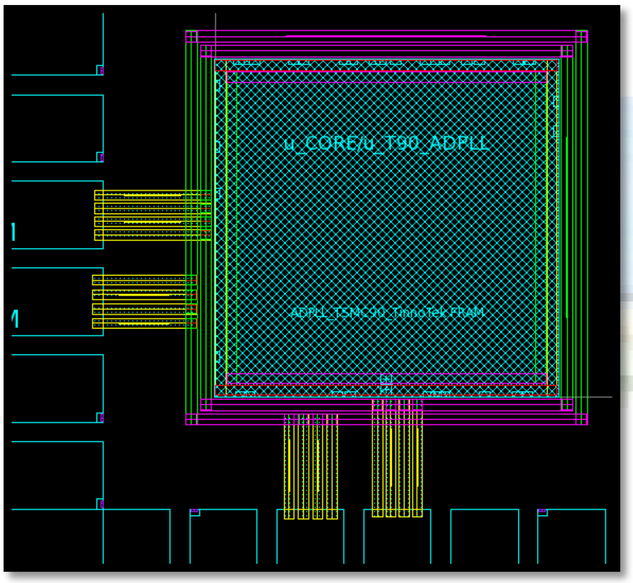
Primary routing layer
○ Preferred: ○ Low [✓ High]
[✓ Pin]
○ Specified: Horizontal layer: M1 (31) Vertical layer: M2 (32)

☐ Customize connection to the pins
Connect to pins at: [✓ Centers] ○ Low ends ○ High ends
Routing width: 0.0 Offset: 0.0

☐ Special rules
Rules:

Route pins on layers
[✓ All] ○ Only pins on layer: M1

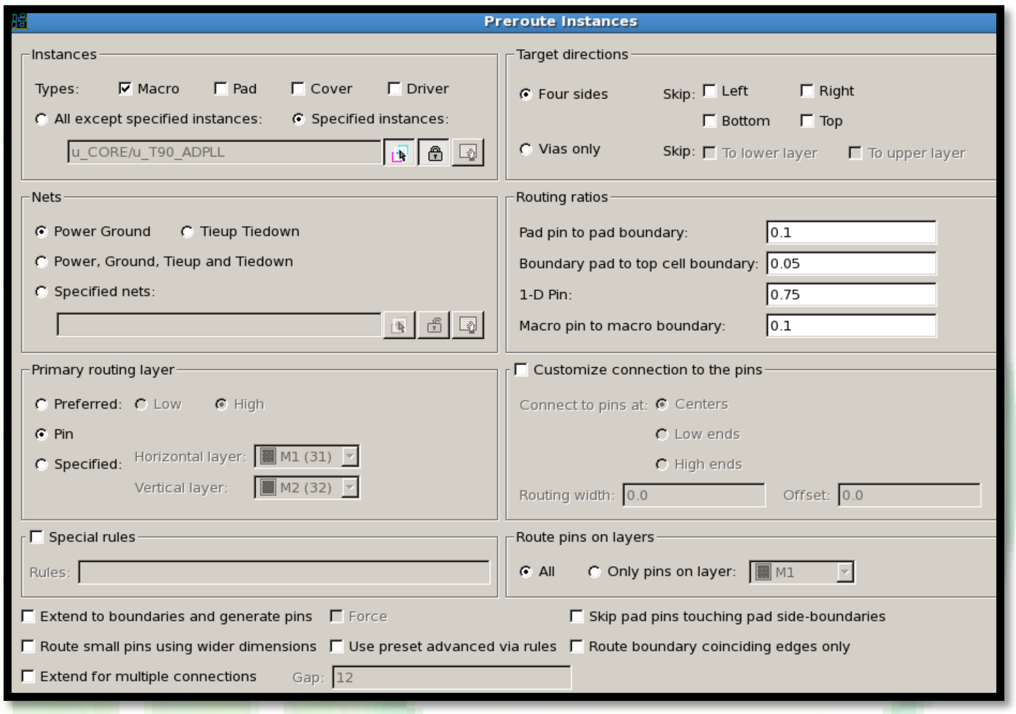
☐ Extend to boundaries and generate pins ☐ Force ☐ Skip pad pins touching pad side-boundaries
☐ Route small pins using wider dimensions ☐ Use preset advanced via rules ☐ Route boundary coinciding edges only
☐ Extend for multiple connections Gap: 12



12. Connect ADPLL PG pin to block ring

Preroute => Preroute Instances…

Type	Macro
Specified instance	check



Preroute Instances

Instances

Types: [✓ Macro] Pad Cover Driver

All except specified instances: [✓ Specified instances:]

u_CORE/u_T90_ADPLL

Target directions

[✓ Four sides] Skip: Left Right Bottom Top

Vias only Skip: To lower layer To upper layer

Nets

[✓ Power Ground] Tieup Tiedown

Power, Ground, Tieup and Tiedown

Specified nets:

Routing ratios

Pad pin to pad boundary:	0.1
Boundary pad to top cell boundary:	0.05
1-D Pin:	0.75
Macro pin to macro boundary:	0.1

Primary routing layer

Preferred: Low [✓ High]

[✓ Pin]

Specified: Horizontal layer: M1 (31) Vertical layer: M2 (32)

Customize connection to the pins

Connect to pins at: [✓ Centers] Low ends High ends

Routing width: 0.0 Offset: 0.0

Special rules

Rules:

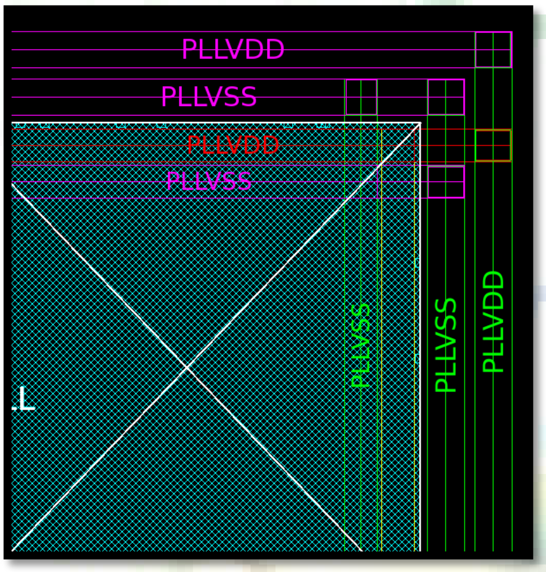
Route pins on layers

[✓ All] Only pins on layer: M1

Extend to boundaries and generate pins Force Skip pad pins touching pad side-boundaries

Route small pins using wider dimensions Use preset advanced via rules Route boundary coinciding edges only

Extend for multiple connections Gap: 12



13. Place other cells and then finish design.

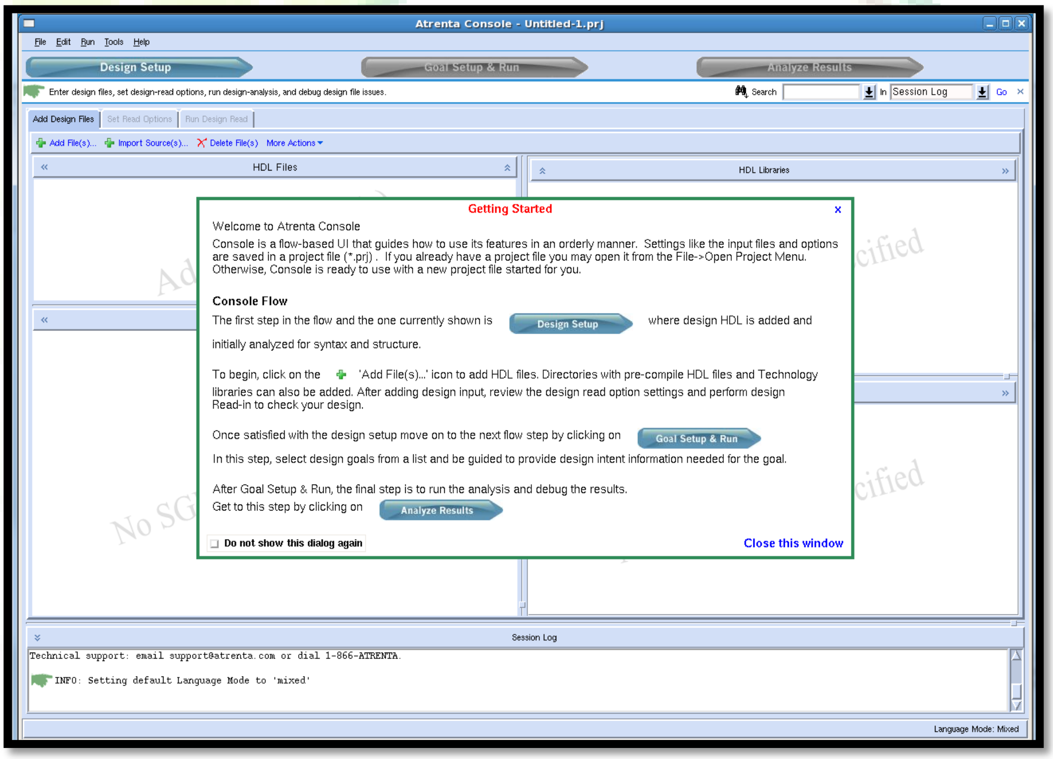
Lab 5 (SKIP) Setup for SpyGlass Constraint File

1. Change directory to lab 5

   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_5

2. Open the SpyGlass in interactive mode

   Unix% spyglass



Enter design files, set design-read options, run design-analysis, and debug design file issues.

Getting Started

Welcome to Atrenta Console

Console is a flow-based UI that guides how to use its features in an orderly manner. Settings like the input files and options are saved in a project file (*.prj) . If you already have a project file you may open it from the File->Open Project Menu. Otherwise, Console is ready to use with a new project file started for you.

Console Flow

The first step in the flow and the one currently shown is Design Setup where design HDL is added and initially analyzed for syntax and structure.

To begin, click on the 'Add File(s)...' icon to add HDL files. Directories with pre-compile HDL files and Technology libraries can also be added. After adding design input, review the design read option settings and perform design Read-in to check your design.

Once satisfied with the design setup move on to the next flow step by clicking on Goal Setup & Run

In this step, select design goals from a list and be guided to provide design intent information needed for the goal.

After Goal Setup & Run, the final step is to run the analysis and debug the results.
Get to this step by clicking on Analyze Results

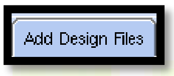
Do not show this dialog again

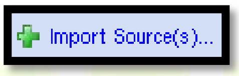
Close this window

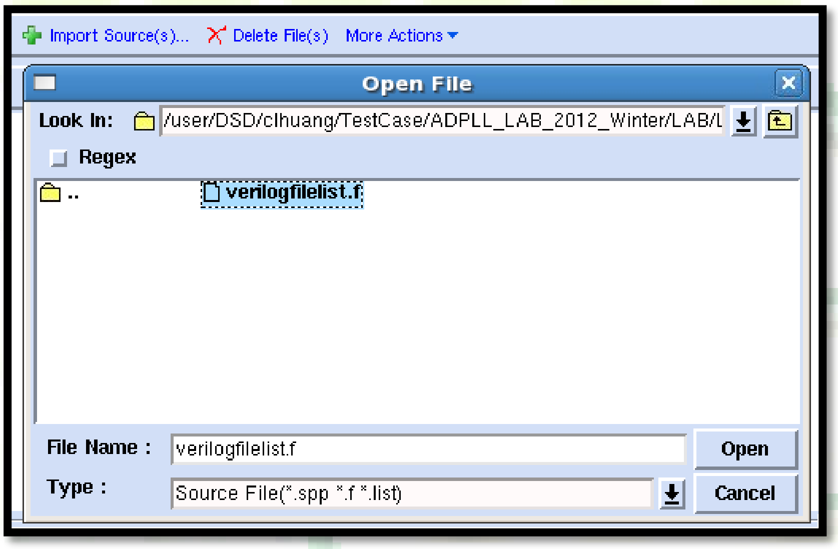
```
Technical support: email support@atrenta.com or dial 1-866-ATRENTA.
INFO: Setting default Language Mode to 'mixed'
```

3. Reading design list

Function tab	Add Design Files
Source list	verilogfilelist.f





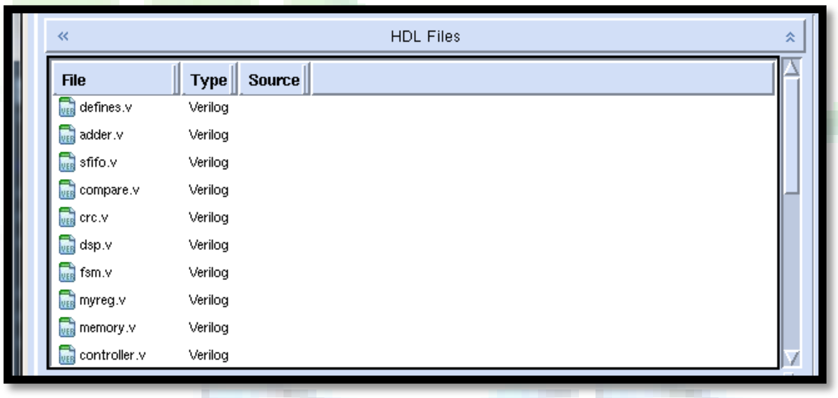


Look In: /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/LAB/L

..
verilogfilelist.f

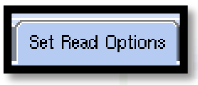
File Name : verilogfilelist.f

Type : Source File(*.spp *.f *.list)



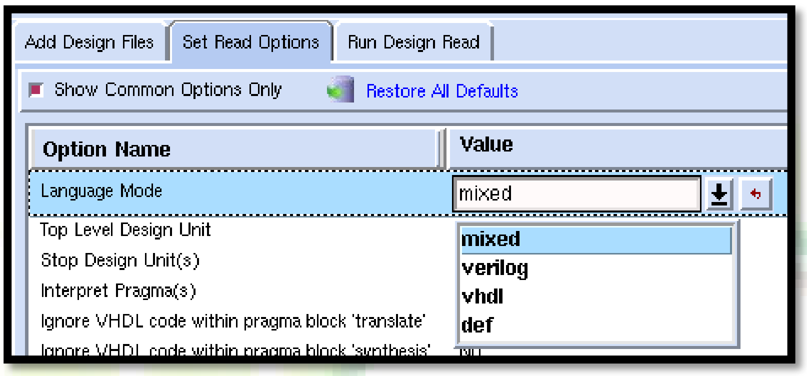
File	Type	Source
defines.v	Verilog	
adder.v	Verilog	
sfifo.v	Verilog	
compare.v	Verilog	
crc.v	Verilog	
dsp.v	Verilog	
fsm.v	Verilog	
myreg.v	Verilog	
memory.v	Verilog	
controller.v	Verilog	

4. Switch function tab to “Set Read Options”



5. Setting language mode to "verilog"

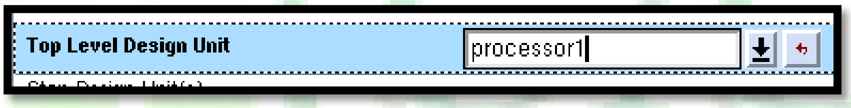
Language Mode	verilog



Option Name	Value
Language Mode	mixed
Top Level Design Unit	
Stop Design Unit(s)	
Interpret Pragma(s)	
Ignore VHDL code within pragma block 'translate'	
Ignore VHDL code within pragma block 'synthesis'	

6. Setting top module name

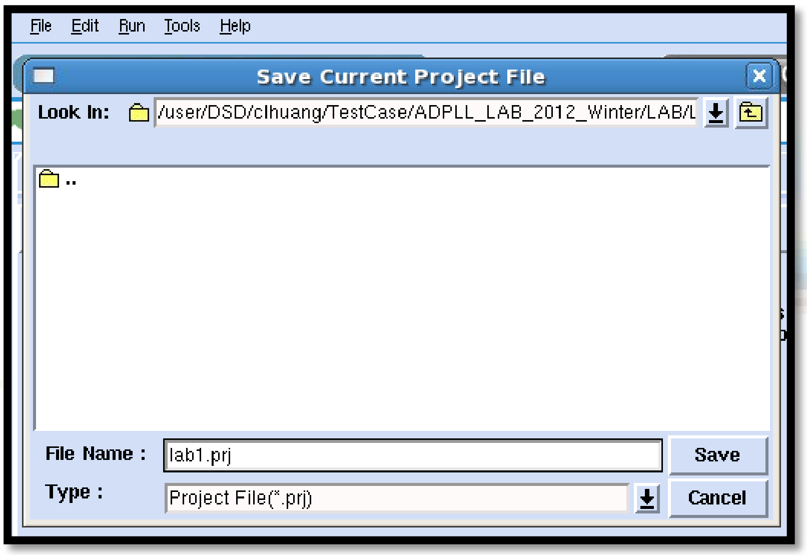
Top level Design Unit	processor1



Top Level Design Unit	processor1

7. Switch function tab to "Run Design Read"

8. Saving project before read design
File => Save Project As => lab5 => Save

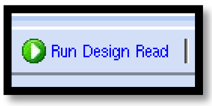


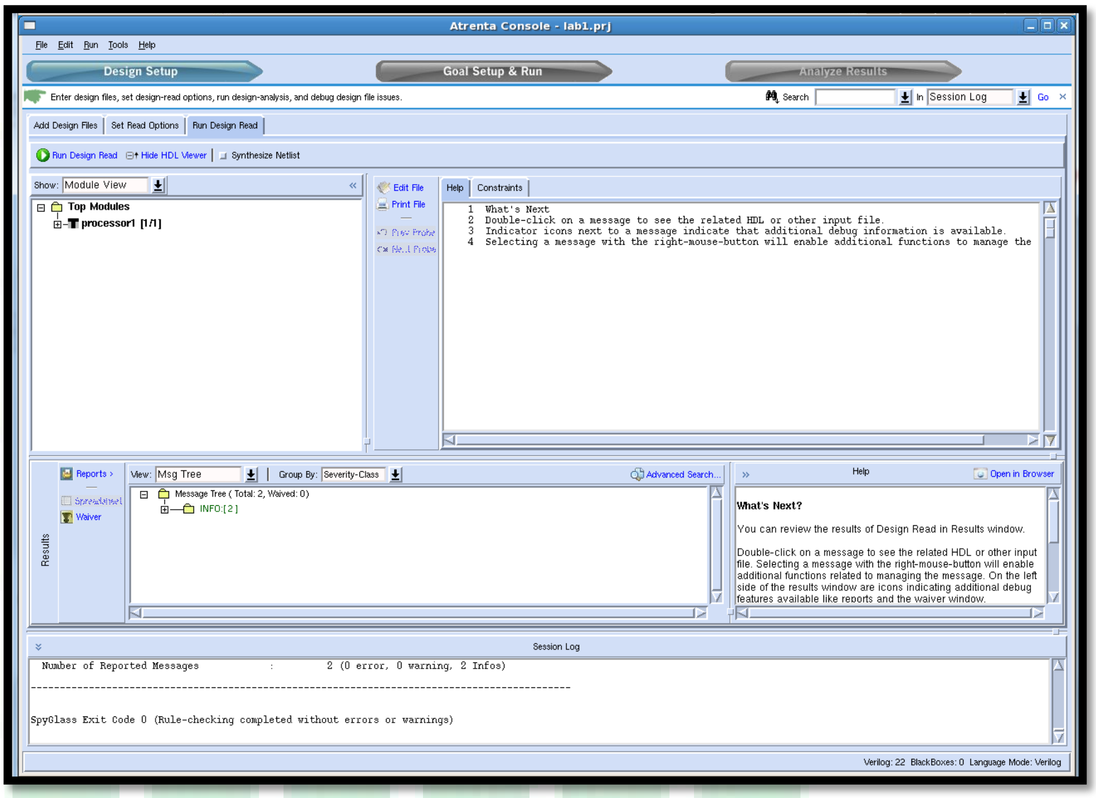
Look In: /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/LAB/L

File Name : lab1.prj

Type : Project File(*.prj)

9. Reading design





Enter design files, set design-read options, run design-analysis, and debug design file issues.

```
What's Next
Double-click on a message to see the related HDL or other input file.
Indicator icons next to a message indicate that additional debug information is available.
Selecting a message with the right-mouse-button will enable additional functions to manage the
```

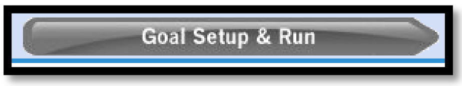
What's Next?

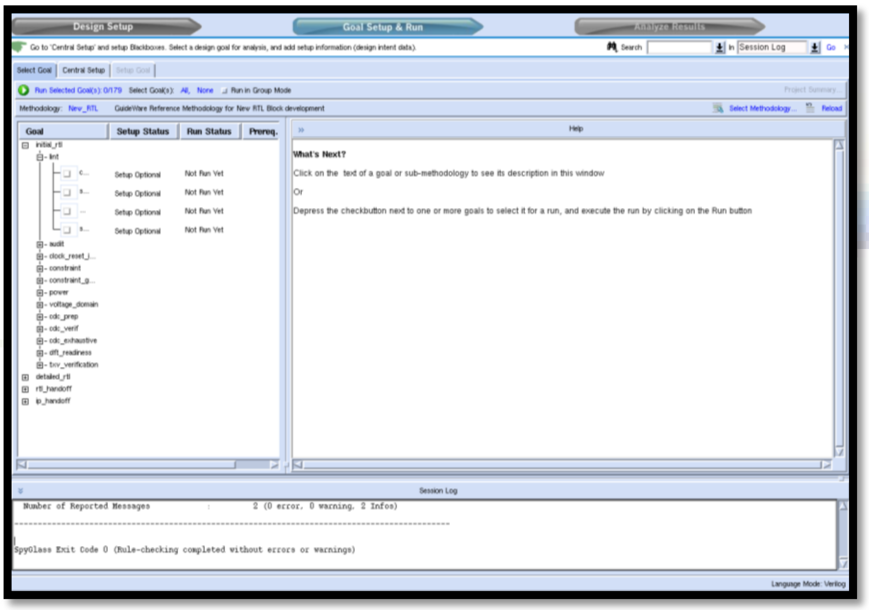
You can review the results of Design Read in Results window.

Double-click on a message to see the related HDL or other input file. Selecting a message with the right-mouse-button will enable additional functions related to managing the message. On the left side of the results window are icons indicating additional debug features available like reports and the waiver window.

```
 Number of Reported Messages           :          2 (0 error, 0 warning, 2 Infos)
-------------------------------------------------------------------------------------

SpyGlass Exit Code 0 (Rule-checking completed without errors or warnings)
```

10. Goal setup





Go to 'Central Setup' and setup Blackboxes. Select a design goal for analysis, and add setup information (design intent data).

Goal	Setup Status	Run Status	Prereq.
initial_rtl			
- lint			
c...	Setup Optional	Not Run Yet	
s...	Setup Optional	Not Run Yet	
...	Setup Optional	Not Run Yet	
s...	Setup Optional	Not Run Yet	
- audit			
- clock_reset_i...			
- constraint			
- constraint_g...			
- power			
- voltage_domain			
- cdc_prep			
- cdc_verif			
- cdc_exhaustive			
- dft_readiness			
- txv_verification			
detailed_rtl			
rtl_handoff			
ip_handoff			

What's Next?

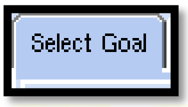
Click on the text of a goal or sub-methodology to see its description in this window

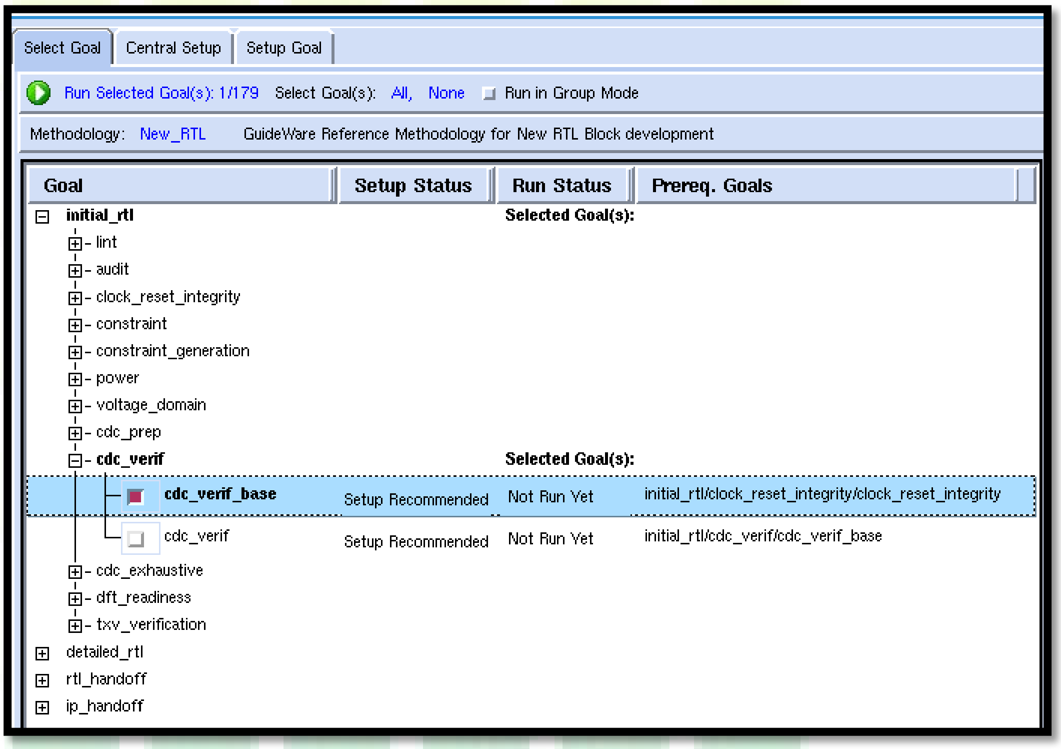
Or

Depress the checkbutton next to one or more goals to select it for a run, and execute the run by clicking on the Run button

```
 Number of Reported Messages           :          2 (0 error, 0 warning, 2 Infos)
-------------------------------------------------------------------------------------

SpyGlass Exit Code 0 (Rule-checking completed without errors or warnings)
```

11. Switch function tab to “Select Goal” and select CDC methodology
    Goal => initial_rtl => cdc_verif => cdc_verif_base

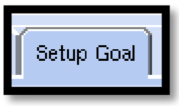


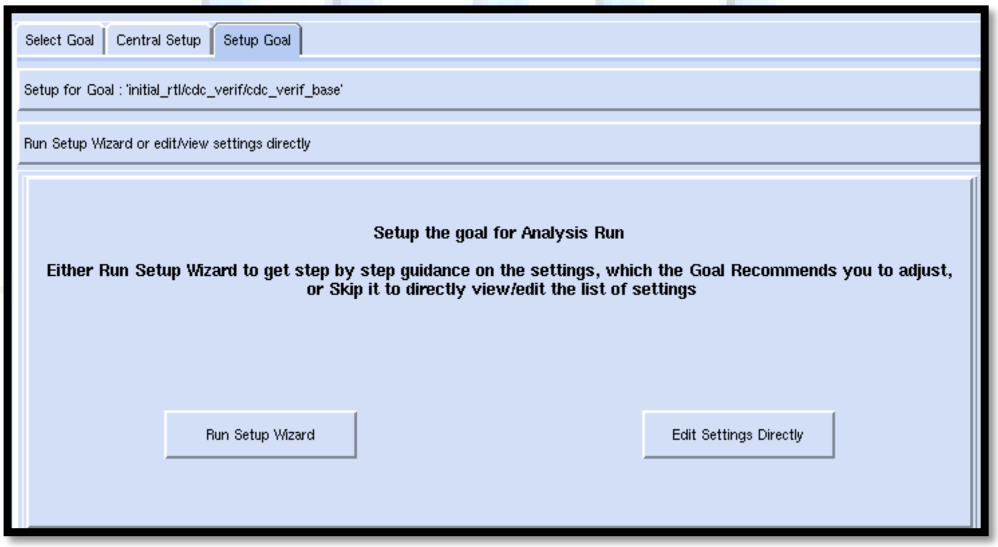


Methodology: New_RTL GuideWare Reference Methodology for New RTL Block development

Goal	Setup Status	Run Status	Prereq. Goals
initial_rtl		Selected Goal(s):	
lint			
audit			
clock_reset_integrity			
constraint			
constraint_generation			
power			
voltage_domain			
cdc_prep			
cdc_verif		Selected Goal(s):	
cdc_verif_base	Setup Recommended	Not Run Yet	initial_rtl/clock_reset_integrity/clock_reset_integrity
cdc_verif	Setup Recommended	Not Run Yet	initial_rtl/cdc_verif/cdc_verif_base
cdc_exhaustive			
dft_readiness			
txv_verification			
detailed_rtl			
rtl_handoff			
ip_handoff			

12. Switch function tab to “Setup Goal” and run setup wizard



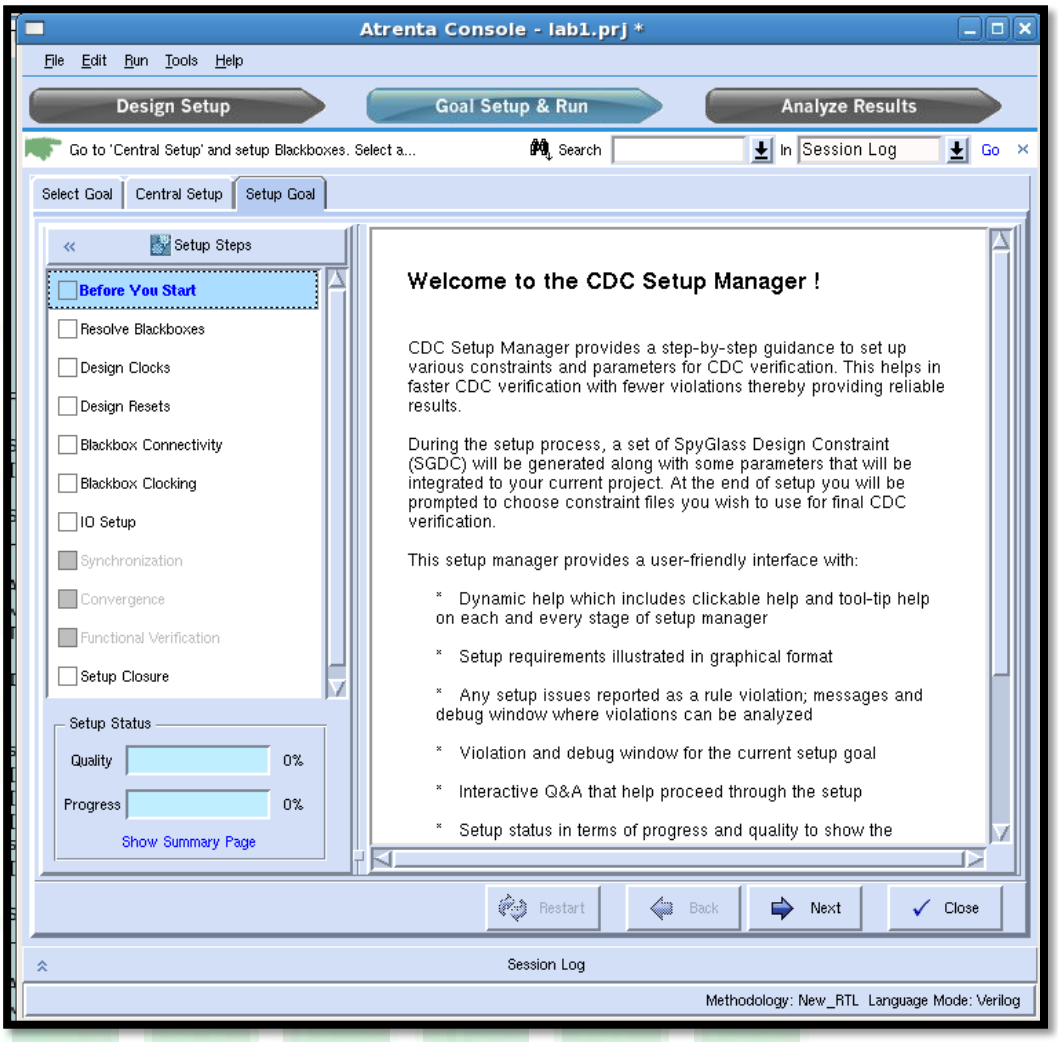


Setup for Goal : 'initial_rtl/cdc_verif/cdc_verif_base'

Run Setup Wizard or edit/view settings directly

Setup the goal for Analysis Run

Either Run Setup Wizard to get step by step guidance on the settings, which the Goal Recommends you to adjust, or Skip it to directly view/edit the list of settings



Welcome to the CDC Setup Manager !

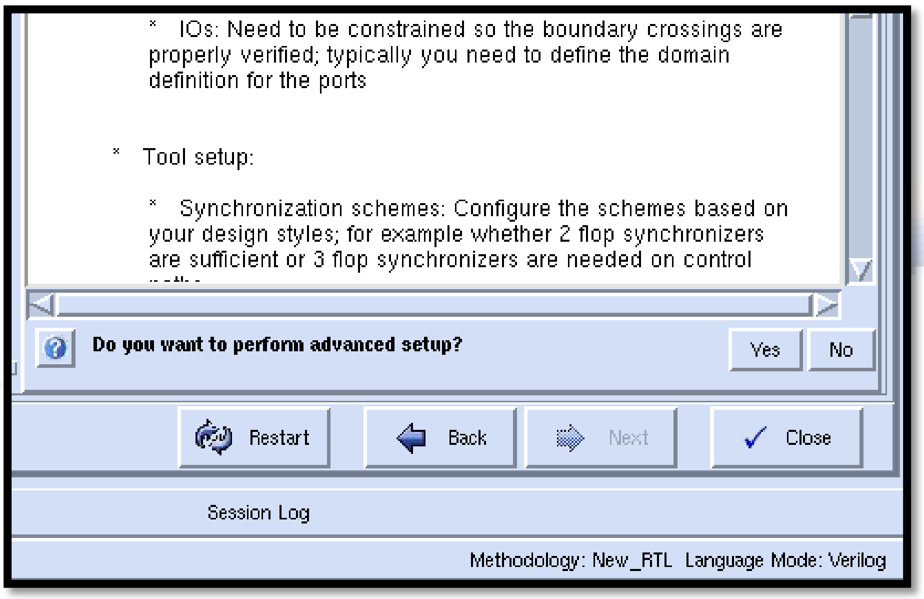
CDC Setup Manager provides a step-by-step guidance to set up various constraints and parameters for CDC verification. This helps in faster CDC verification with fewer violations thereby providing reliable results.

During the setup process, a set of SpyGlass Design Constraint (SGDC) will be generated along with some parameters that will be integrated to your current project. At the end of setup you will be prompted to choose constraint files you wish to use for final CDC verification.

This setup manager provides a user-friendly interface with:

* Dynamic help which includes clickable help and tool-tip help on each and every stage of setup manager
* Setup requirements illustrated in graphical format
* Any setup issues reported as a rule violation; messages and debug window where violations can be analyzed
* Violation and debug window for the current setup goal
* Interactive Q&A that help proceed through the setup
* Setup status in terms of progress and quality to show the

13. Click “Next” and click “No” to run the basic setup.


* IOs: Need to be constrained so the boundary crossings are properly verified; typically you need to define the domain definition for the ports

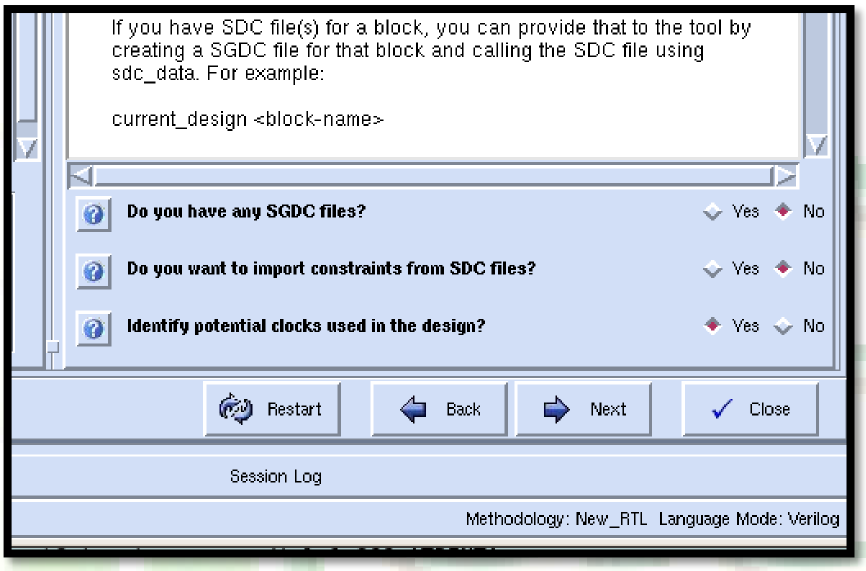
* Tool setup:

  * Synchronization schemes: Configure the schemes based on your design styles; for example whether 2 flop synchronizers are sufficient or 3 flop synchronizers are needed on control

Do you want to perform advanced setup?

14. Let SpyGlass identify everything!

Do you have any SGDC files	No
Do you want to import constraints from SDC files?	No
Identify potential clocks used in the design?	**Yes**


If you have SDC file(s) for a block, you can provide that to the tool by creating a SGDC file for that block and calling the SDC file using sdc_data. For example:

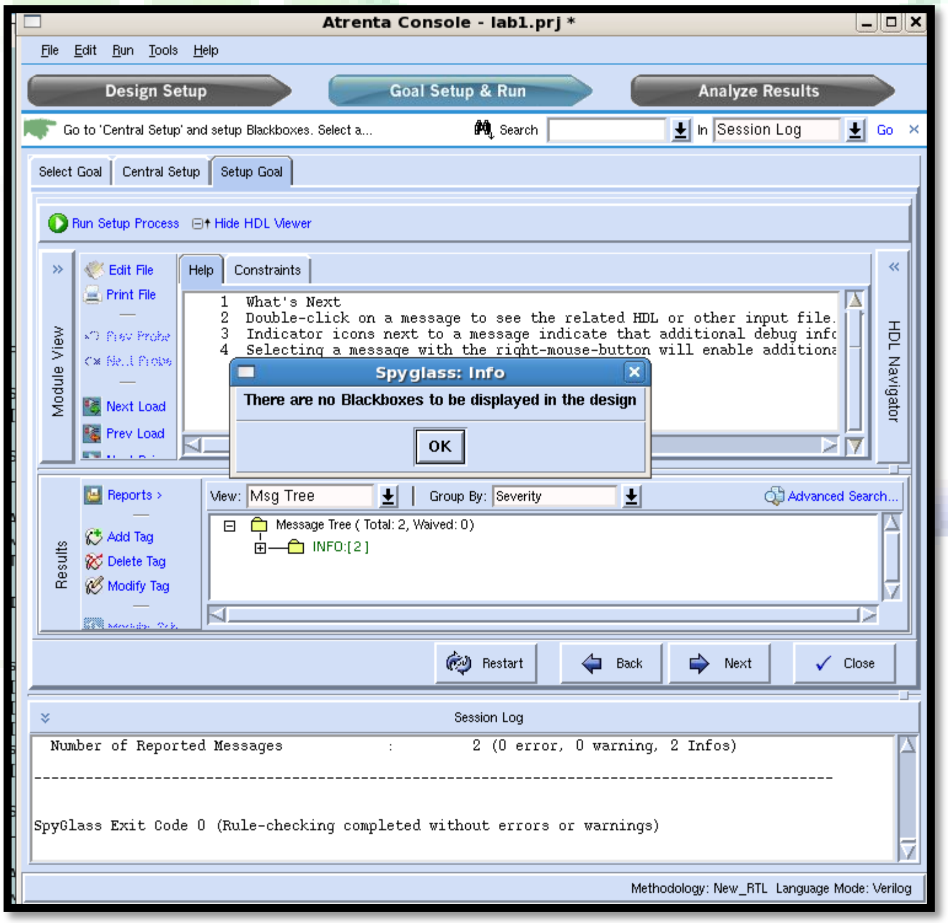
current_design <block-name>

Do you have any SGDC files? Yes [✓ No]

Do you want to import constraints from SDC files? Yes [✓ No]

Identify potential clocks used in the design? [✓ Yes] No

15. Resolve blackboxes



```
What's Next
Double-click on a message to see the related HDL or other input file.
Indicator icons next to a message indicate that additional debug info
Selecting a message with the right-mouse-button will enable additiona
```

Spyglass: Info

There are no Blackboxes to be displayed in the design

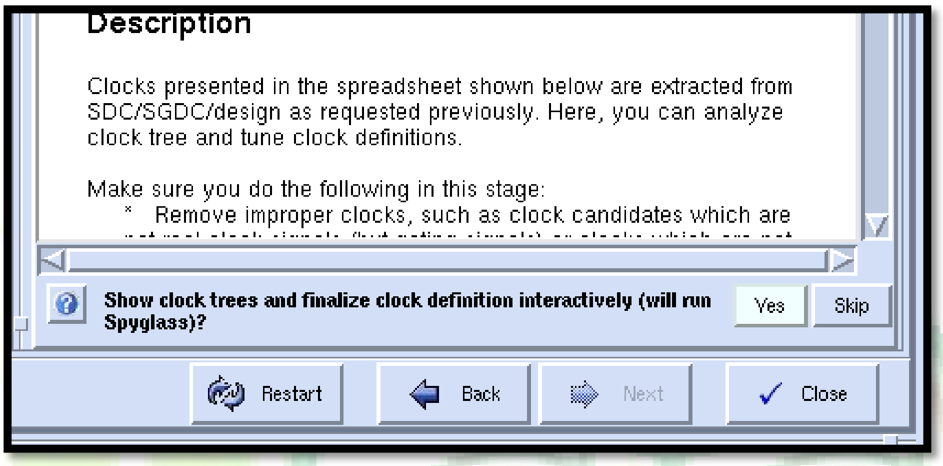
OK

```
Message Tree ( Total: 2, Waived: 0)
    INFO:[2]
```

```
Number of Reported Messages          :       2 (0 error, 0 warning, 2 Infos)
------------------------------------------------------------------------------------

SpyGlass Exit Code 0 (Rule-checking completed without errors or warnings)
```

16. Setting clocks

Show clock trees and finalize clock definition interactively?	**Yes**

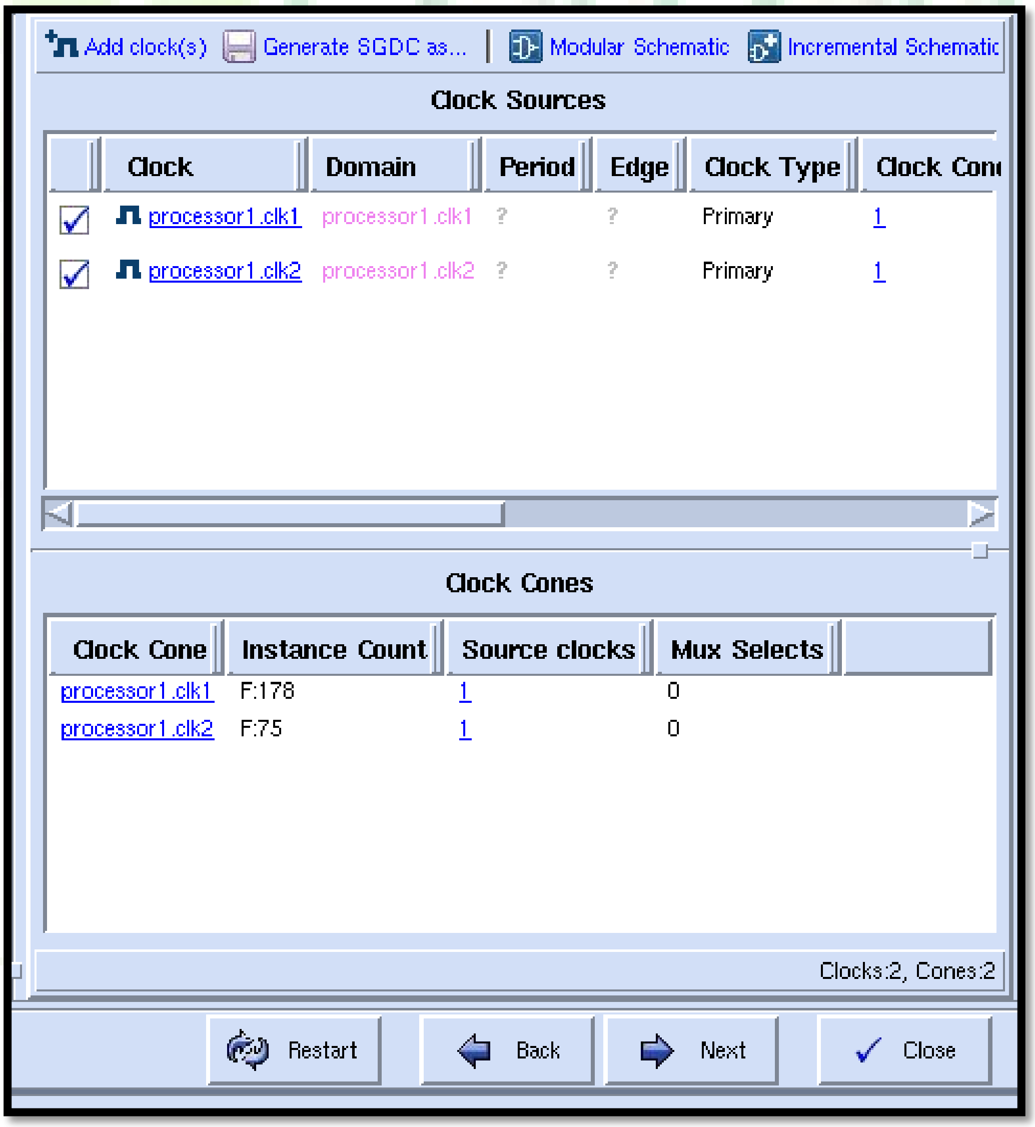


Description

Clocks presented in the spreadsheet shown below are extracted from SDC/SGDC/design as requested previously. Here, you can analyze clock tree and tune clock definitions.

Make sure you do the following in this stage:
* Remove improper clocks, such as clock candidates which are

Show clock trees and finalize clock definition interactively (will run Spyglass)?



Clock Sources

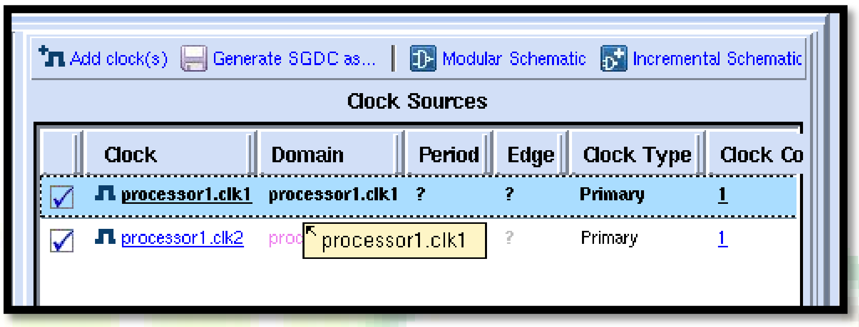
	Clock	Domain	Period	Edge	Clock Type	Clock Con
✓	processor1.clk1	processor1.clk1	?	?	Primary	1
✓	processor1.clk2	processor1.clk2	?	?	Primary	1

Clock Cones

Clock Cone	Instance Count	Source clocks	Mux Selects
processor1.clk1	F:178	1	0
processor1.clk2	F:75	1	0

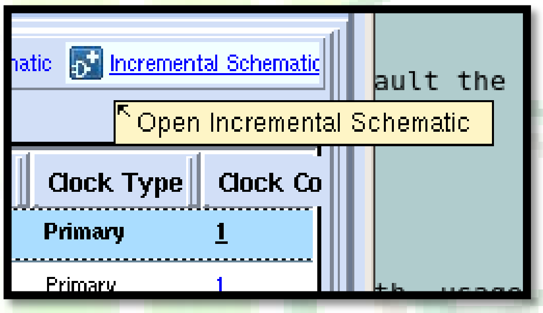
Clocks:2, Cones:2

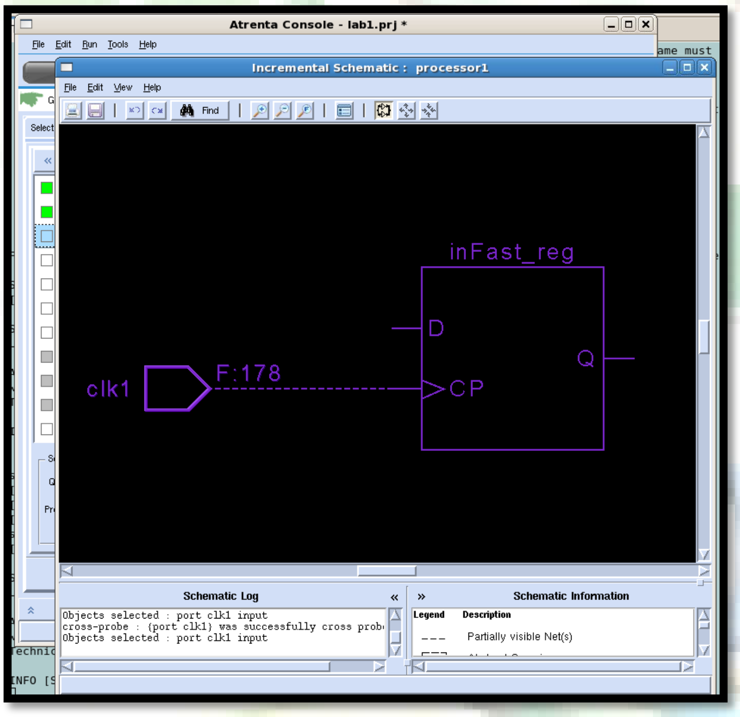
17. Click clock “processor1.clk1” first and then click “incremental schematic”, you can see simplified schematic.



Clock Sources

	Clock	Domain	Period	Edge	Clock Type	Clock Co
✓	processor1.clk1	processor1.clk1	?	?	Primary	1
✓	processor1.clk2	proc		?	Primary	1

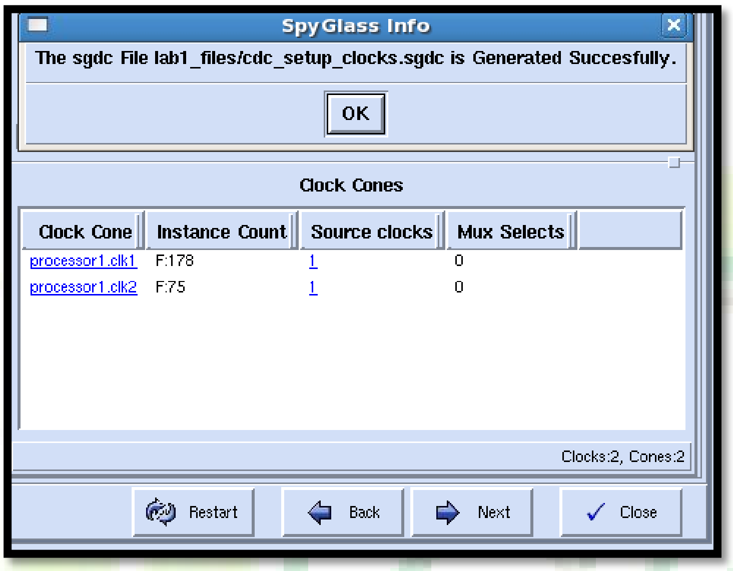




```
Objects selected : port clk1 input
cross-probe : (port clk1) was successfully cross prob
Objects selected : port clk1 input
```

Legend	Description
---	Partially visible Net(s)

18. Click “Next” to generate sgdc file



The sgdc File lab1_files/cdc_setup_clocks.sgdc is Generated Succesfully.

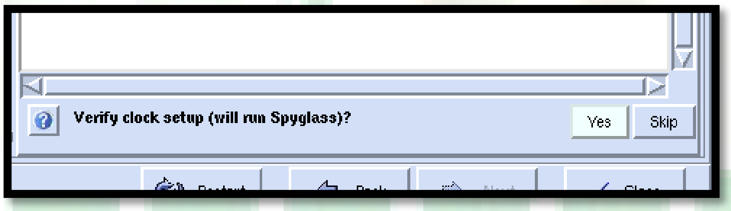
Clock Cones

Clock Cone	Instance Count	Source clocks	Mux Selects
processor1.clk1	F:178	1	0
processor1.clk2	F:75	1	0

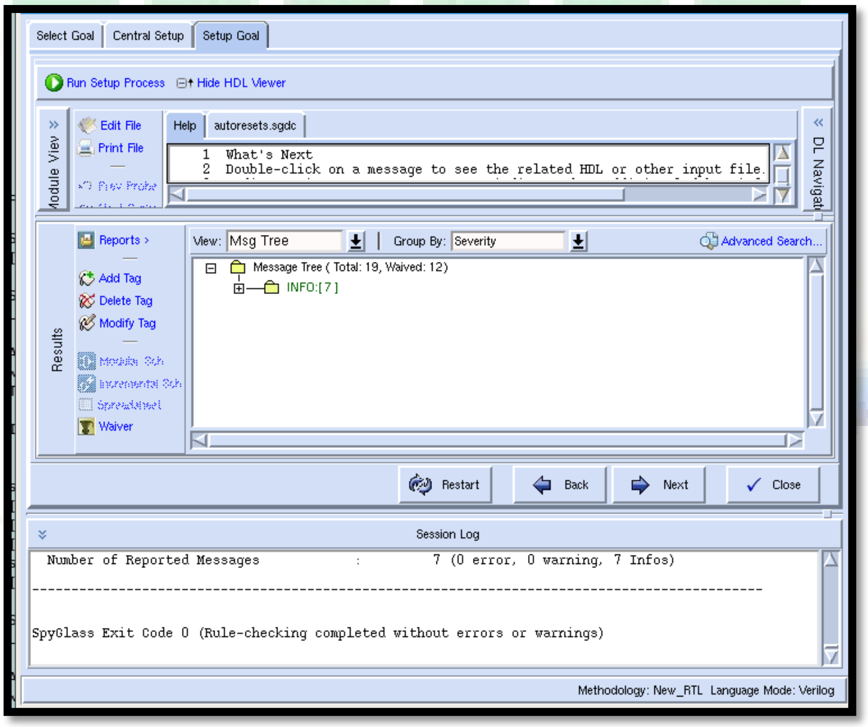
Clocks:2, Cones:2

19. Validate clock constraints

Verify clock setup?	Yes



Verify clock setup (will run Spyglass)?



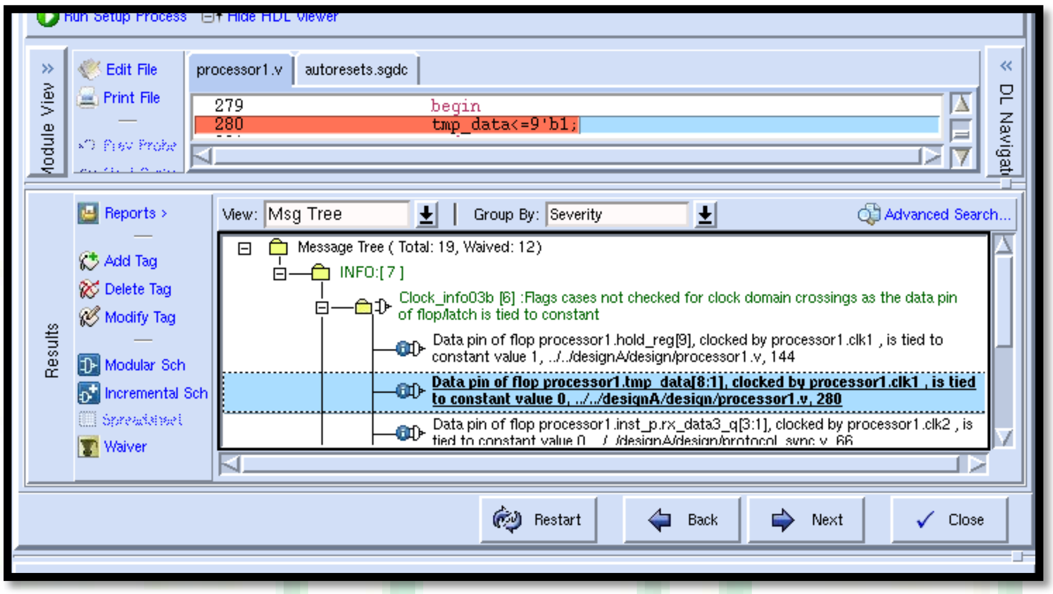
```
What's Next
Double-click on a message to see the related HDL or other input file.
```

Message Tree (Total: 19, Waived: 12)
INFO:[7]

```
Number of Reported Messages          :          7 (0 error, 0 warning, 7 Infos)
------------------------------------------------------------------------------------

SpyGlass Exit Code 0 (Rule-checking completed without errors or warnings)
```

Methodology: New_RTL Language Mode: Verilog

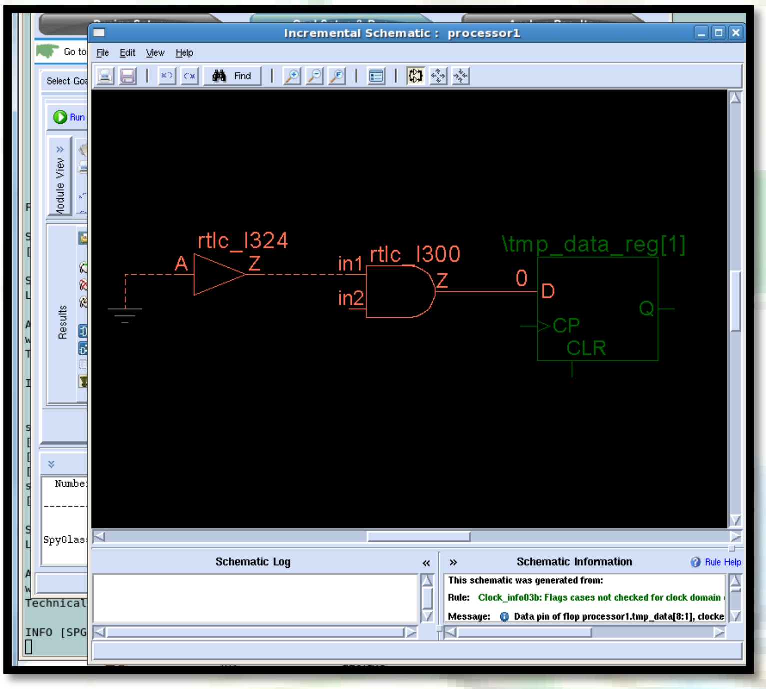
20. You can see some information at “Message Tree”



```
279                    begin
280                    tmp_data<=9'b1;
```

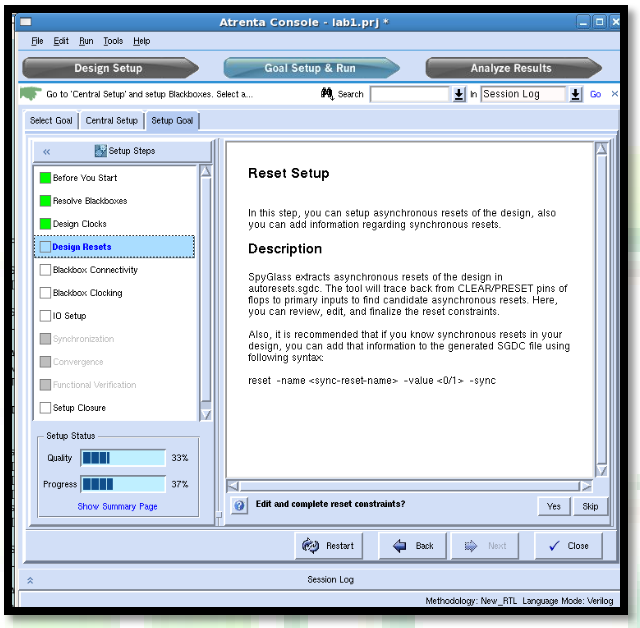
Message Tree (Total: 19, Waived: 12)
INFO:[7]
Clock_info03b [6] :Flags cases not checked for clock domain crossings as the data pin of flop/latch is tied to constant
Data pin of flop processor1.hold_reg[9], clocked by processor1.clk1 , is tied to constant value 1 , ../../designA/design/processor1.v, 144
Data pin of flop processor1.tmp_data[8:1], clocked by processor1.clk1 , is tied to constant value 0 , ../../designA/design/processor1.v, 280
Data pin of flop processor1.inst_p.rx_data3_q[3:1], clocked by processor1.clk2 , is tied to constant value 0 , ../../designA/design/protocol_sync.v, 66

21. Select one of messages and click “Incremental Sch” to see simplified schematic.



This schematic was generated from:
Rule: Clock_info03b: Flags cases not checked for clock domain
Message: Data pin of flop processor1.tmp_data[8:1], clocke

22. Click “Next” to next setup.



Reset Setup

In this step, you can setup asynchronous resets of the design, also you can add information regarding synchronous resets.

Description

SpyGlass extracts asynchronous resets of the design in autoresets.sgdc. The tool will trace back from CLEAR/PRESET pins of flops to primary inputs to find candidate asynchronous resets. Here, you can review, edit, and finalize the reset constraints.

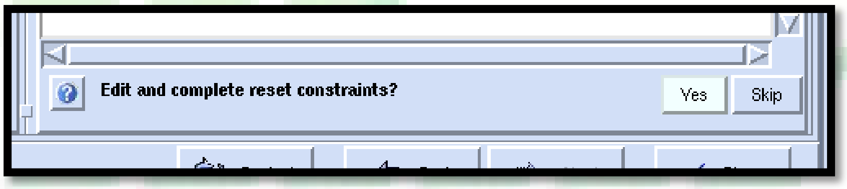
Also, it is recommended that if you know synchronous resets in your design, you can add that information to the generated SGDC file using following syntax:

```
reset -name <sync-reset-name> -value <0/1> -sync
```

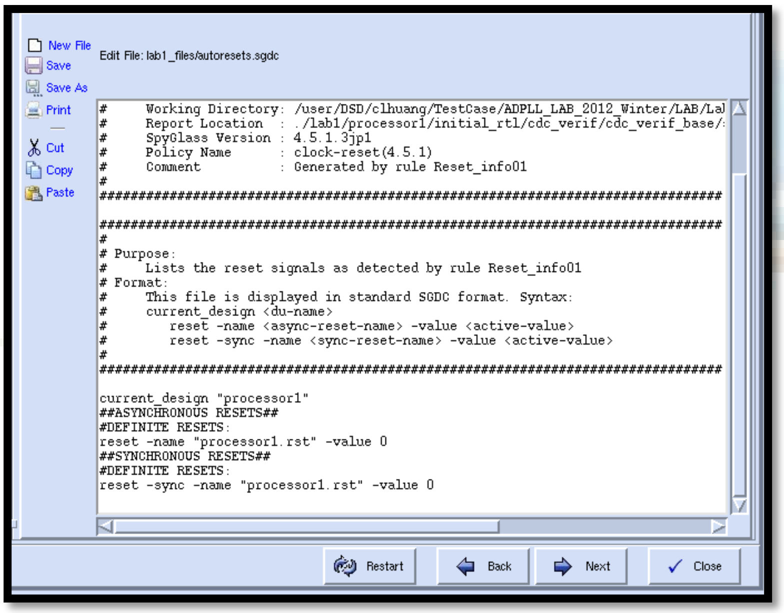
Edit and complete reset constraints?

23. Reset setup

Edit and complete reset constraints?	**Yes**



24. Verify reset is set correctly.

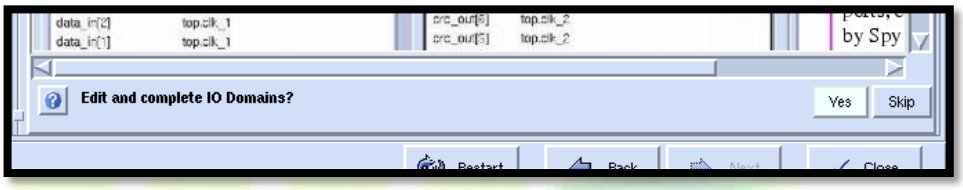


Edit File: lab1_files/autoresets.sgdc

```
#     Working Directory:  /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/LAB/Lal
#     Report Location  :  ./lab1/processor1/initial_rtl/cdc_verif/cdc_verif_base/:
#     SpyGlass Version :  4.5.1.3jp1
#     Policy Name      :  clock-reset(4.5.1)
#     Comment          :  Generated by rule Reset_info01
#
################################################################################

################################################################################
#
# Purpose:
#     Lists the reset signals as detected by rule Reset_info01
# Format:
#     This file is displayed in standard SGDC format. Syntax:
#     current_design <du-name>
#        reset -name <async-reset-name> -value <active-value>
#        reset -sync -name <sync-reset-name> -value <active-value>
#
################################################################################

current_design "processor1"
##ASYNCHRONOUS RESETS##
#DEFINITE RESETS:
reset -name "processor1.rst" -value 0
##SYNCHRONOUS RESETS##
#DEFINITE RESETS:
reset -sync -name "processor1.rst" -value 0
```

25. IO port and clock domain relationship setup

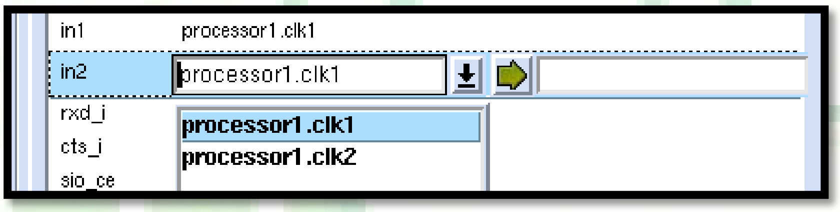
Edit and complete IO domains?	**Yes**



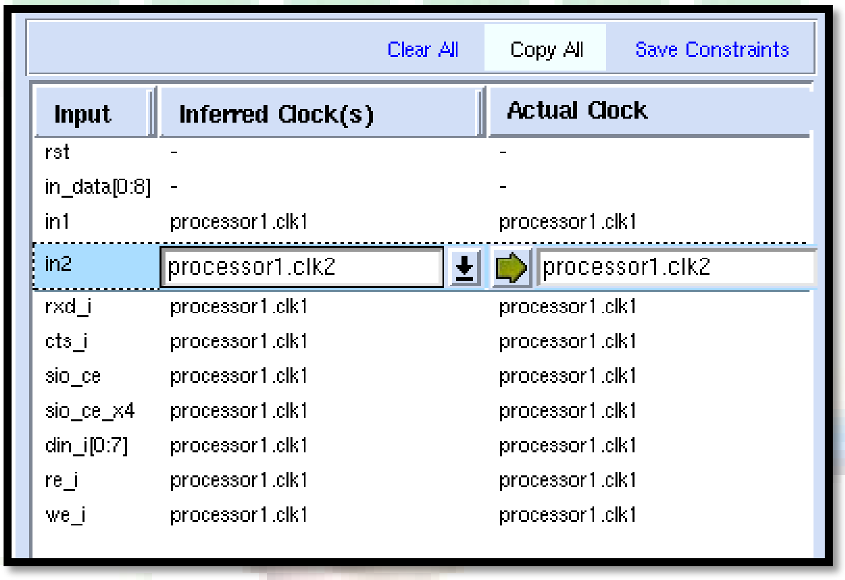
26. Setting actual clock

in2	**processor1.clk2**
other input	processor1.clk1

Click in2 and then select processor1.clk2



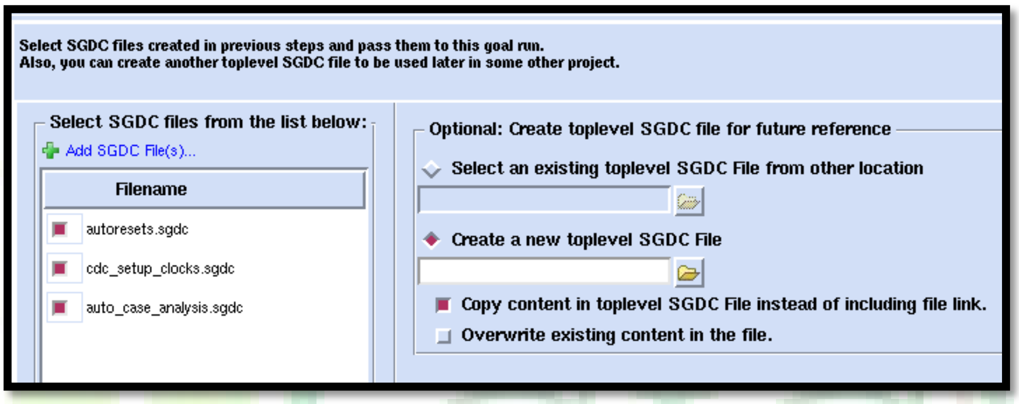
Click “Copy all” to copy all inferred clocks to actual clock



Input	Inferred Clock(s)	Actual Clock
rst	-	-
in_data[0:8]	-	-
in1	processor1.clk1	processor1.clk1
in2	processor1.clk2	processor1.clk2
rxd_i	processor1.clk1	processor1.clk1
cts_i	processor1.clk1	processor1.clk1
sio_ce	processor1.clk1	processor1.clk1
sio_ce_x4	processor1.clk1	processor1.clk1
din_i[0:7]	processor1.clk1	processor1.clk1
re_i	processor1.clk1	processor1.clk1
we_i	processor1.clk1	processor1.clk1

27. Click “Next” move to SGDC file save setting.

Copy content in toplevel SGDC file instead of including file link	**Check**
Other	Default



Select SGDC files created in previous steps and pass them to this goal run.
Also, you can create another toplevel SGDC file to be used later in some other project.

Select SGDC files from the list below:

Add SGDC File(s)...

Filename
🟥 autoresets.sgdc
🟥 cdc_setup_clocks.sgdc
🟥 auto_case_analysis.sgdc

Optional: Create toplevel SGDC file for future reference

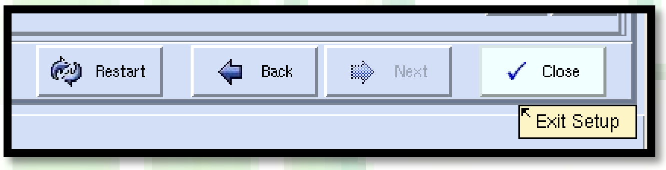
◇ Select an existing toplevel SGDC File from other location

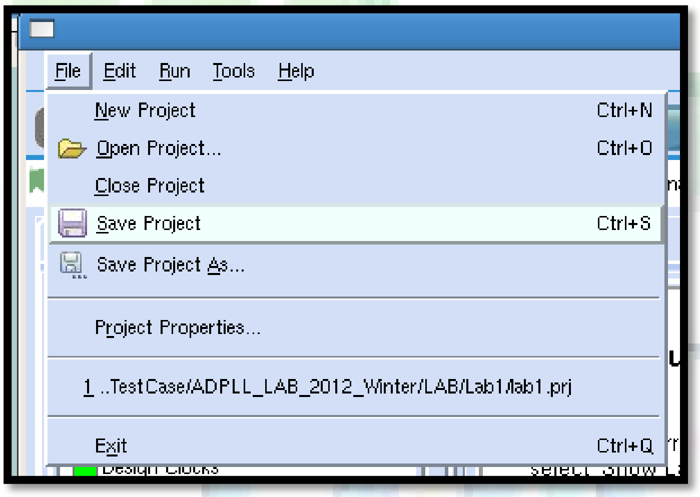
◆ Create a new toplevel SGDC File

🟥 Copy content in toplevel SGDC File instead of including file link.

☐ Overwrite existing content in the file.

28. Click “Close” and save project





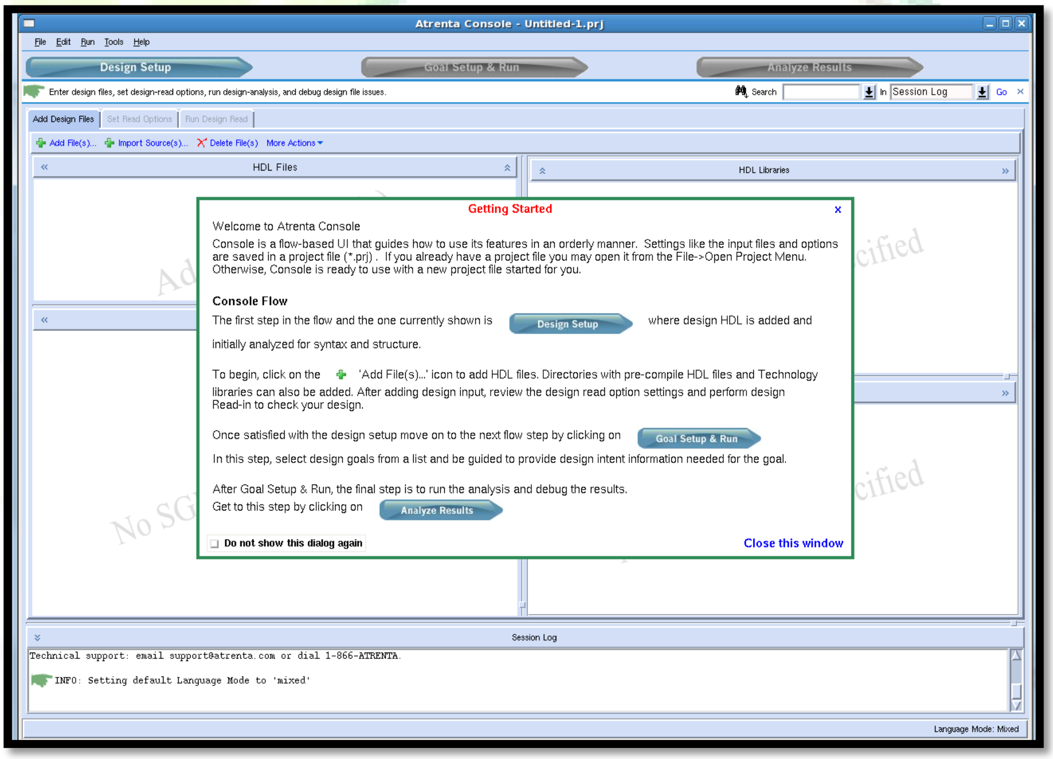
Lab 6 (SKIP) Clock/Reset integrity check

1. Change directory to lab 6

   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_6

2. Open the SpyGlass in interactive mode

   Unix% spyglass



Enter design files, set design-read options, run design-analysis, and debug design file issues.

Getting Started

Welcome to Atrenta Console

Console is a flow-based UI that guides how to use its features in an orderly manner. Settings like the input files and options are saved in a project file (*.prj) . If you already have a project file you may open it from the File->Open Project Menu. Otherwise, Console is ready to use with a new project file started for you.

Console Flow

The first step in the flow and the one currently shown is Design Setup where design HDL is added and initially analyzed for syntax and structure.

To begin, click on the 'Add File(s)...' icon to add HDL files. Directories with pre-compile HDL files and Technology libraries can also be added. After adding design input, review the design read option settings and perform design Read-in to check your design.

Once satisfied with the design setup move on to the next flow step by clicking on Goal Setup & Run

In this step, select design goals from a list and be guided to provide design intent information needed for the goal.

After Goal Setup & Run, the final step is to run the analysis and debug the results.
Get to this step by clicking on Analyze Results

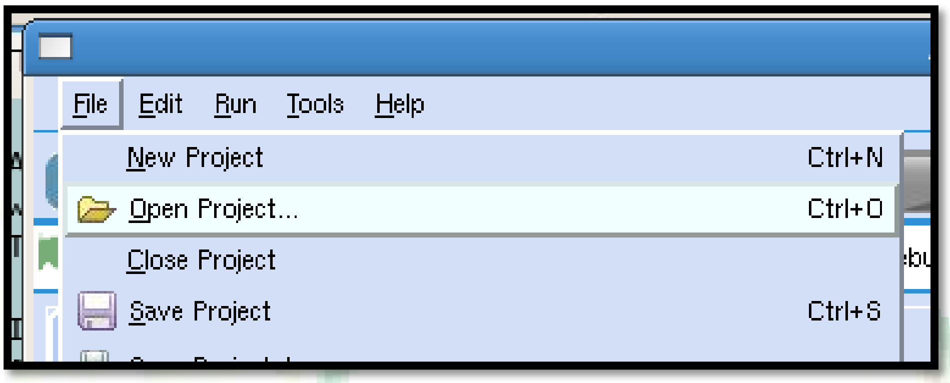
Do not show this dialog again

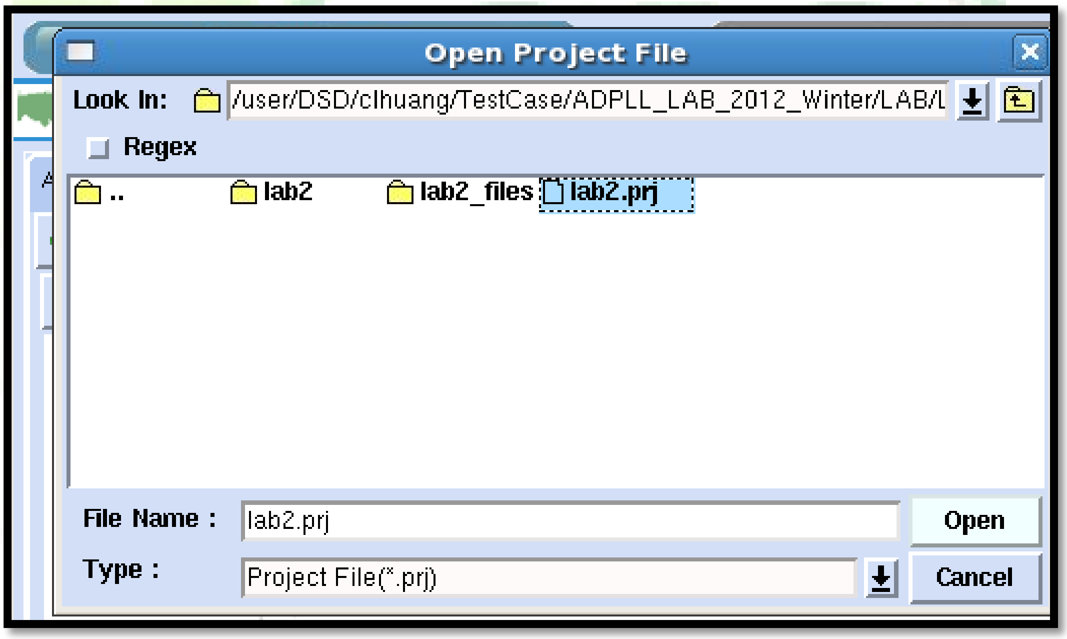
Close this window

```
Technical support: email support@atrenta.com or dial 1-866-ATRENTA.
INFO: Setting default Language Mode to 'mixed'
```

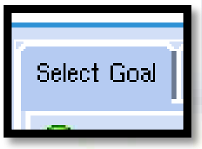
3. Loading project

Project file name	Lab6.prj




Open Project File
Look In: /user/DSD/clhuang/TestCase/ADPLL_LAB_2012_Winter/LAB/L
Regex
..
lab2
lab2_files
lab2.prj
File Name : lab2.prj
Type : Project File(*.prj)
Open
Cancel

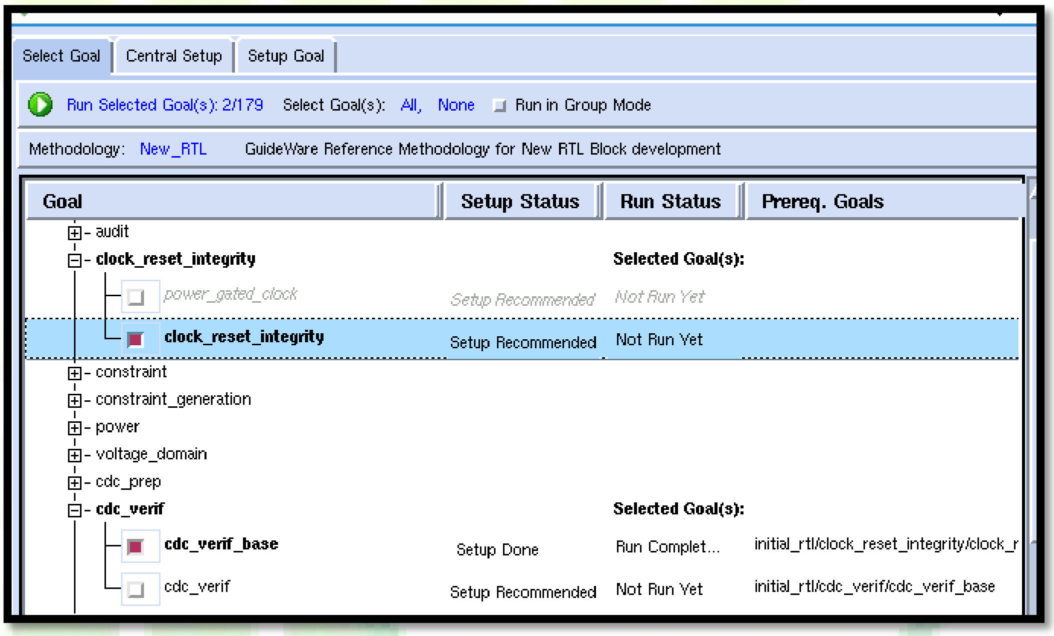
4. Switch function tab to "Select Goal"



5. Select new goal "clock_reset_integrity"

   Goal list

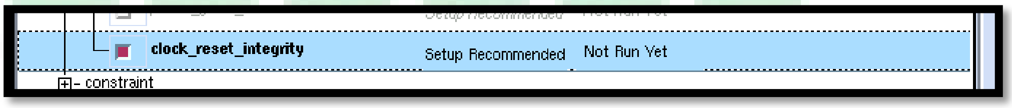
clock_reset_integrity/clock_reset_integrity	**Check**
cdc_verif/cdc_verif_base	**Check**
Others	uncheck



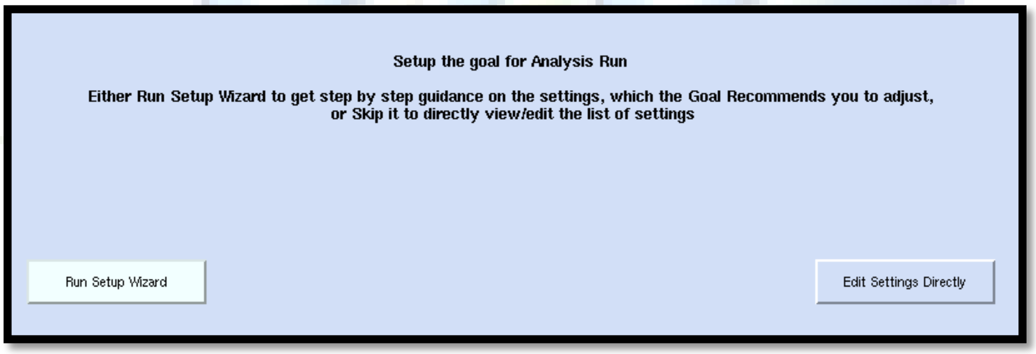
Methodology: New_RTL GuideWare Reference Methodology for New RTL Block development

Goal	Setup Status	Run Status	Prereq. Goals
audit			
clock_reset_integrity		Selected Goal(s):	
power_gated_clock	Setup Recommended	Not Run Yet	
clock_reset_integrity	Setup Recommended	Not Run Yet	
constraint			
constraint_generation			
power			
voltage_domain			
cdc_prep			
cdc_verif		Selected Goal(s):	
cdc_verif_base	Setup Done	Run Complet...	initial_rtl/clock_reset_integrity/clock_r
cdc_verif	Setup Recommended	Not Run Yet	initial_rtl/cdc_verif/cdc_verif_base

6. Highlight the "clock_reset_integrity" goal and switch function tab to "Setup Goal"



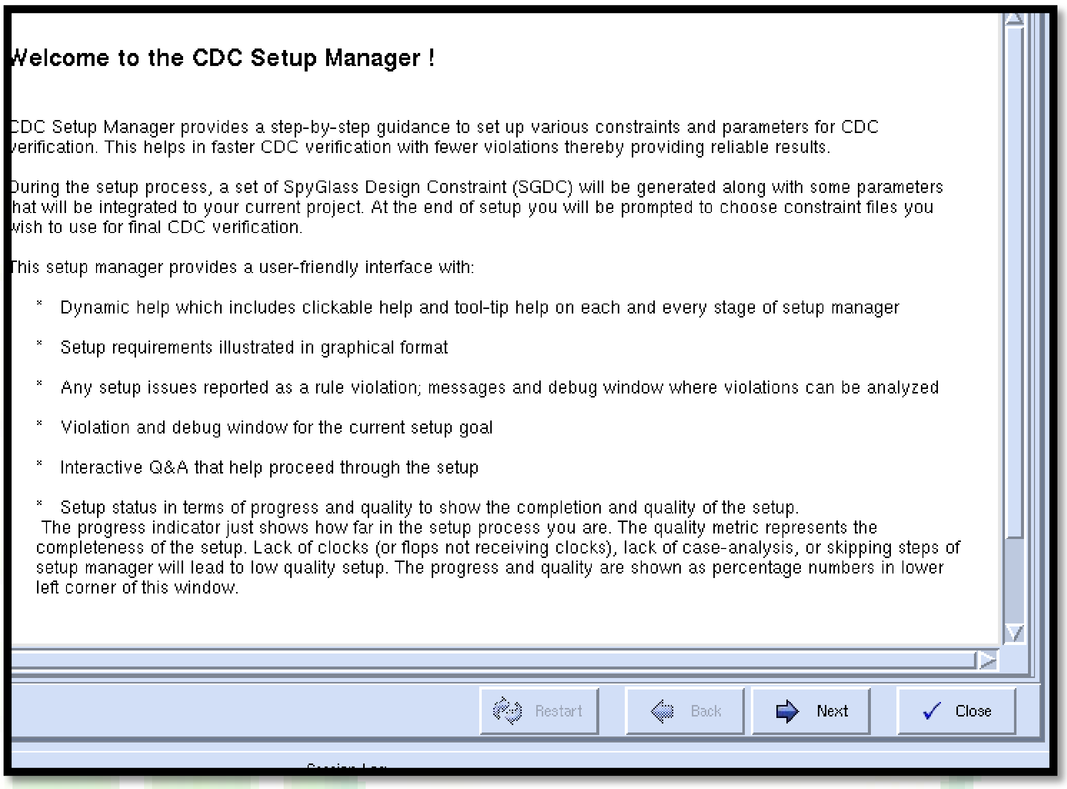
7. Run Setup Wizard for setup "clock_reset_integrity" goal



Setup the goal for Analysis Run

Either Run Setup Wizard to get step by step guidance on the settings, which the Goal Recommends you to adjust, or Skip it to directly view/edit the list of settings

8. Click “Next” to next step


Welcome to the CDC Setup Manager !

CDC Setup Manager provides a step-by-step guidance to set up various constraints and parameters for CDC verification. This helps in faster CDC verification with fewer violations thereby providing reliable results.

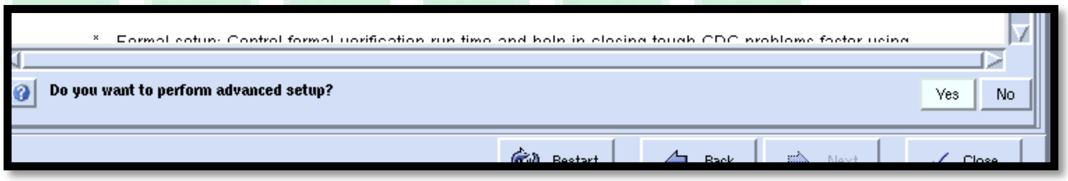
During the setup process, a set of SpyGlass Design Constraint (SGDC) will be generated along with some parameters that will be integrated to your current project. At the end of setup you will be prompted to choose constraint files you wish to use for final CDC verification.

This setup manager provides a user-friendly interface with:

* Dynamic help which includes clickable help and tool-tip help on each and every stage of setup manager
* Setup requirements illustrated in graphical format
* Any setup issues reported as a rule violation; messages and debug window where violations can be analyzed
* Violation and debug window for the current setup goal
* Interactive Q&A that help proceed through the setup
* Setup status in terms of progress and quality to show the completion and quality of the setup. The progress indicator just shows how far in the setup process you are. The quality metric represents the completeness of the setup. Lack of clocks (or flops not receiving clocks), lack of case-analysis, or skipping steps of setup manager will lead to low quality setup. The progress and quality are shown as percentage numbers in lower left corner of this window.

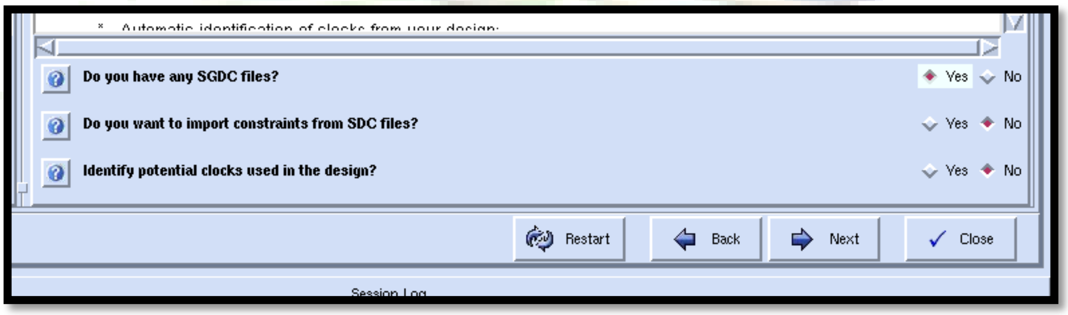
9. Perform basic setup

Do you want to perform advance setup?	**No**


Do you want to perform advanced setup? Yes No

10. Let SpyGlass identify everthing!

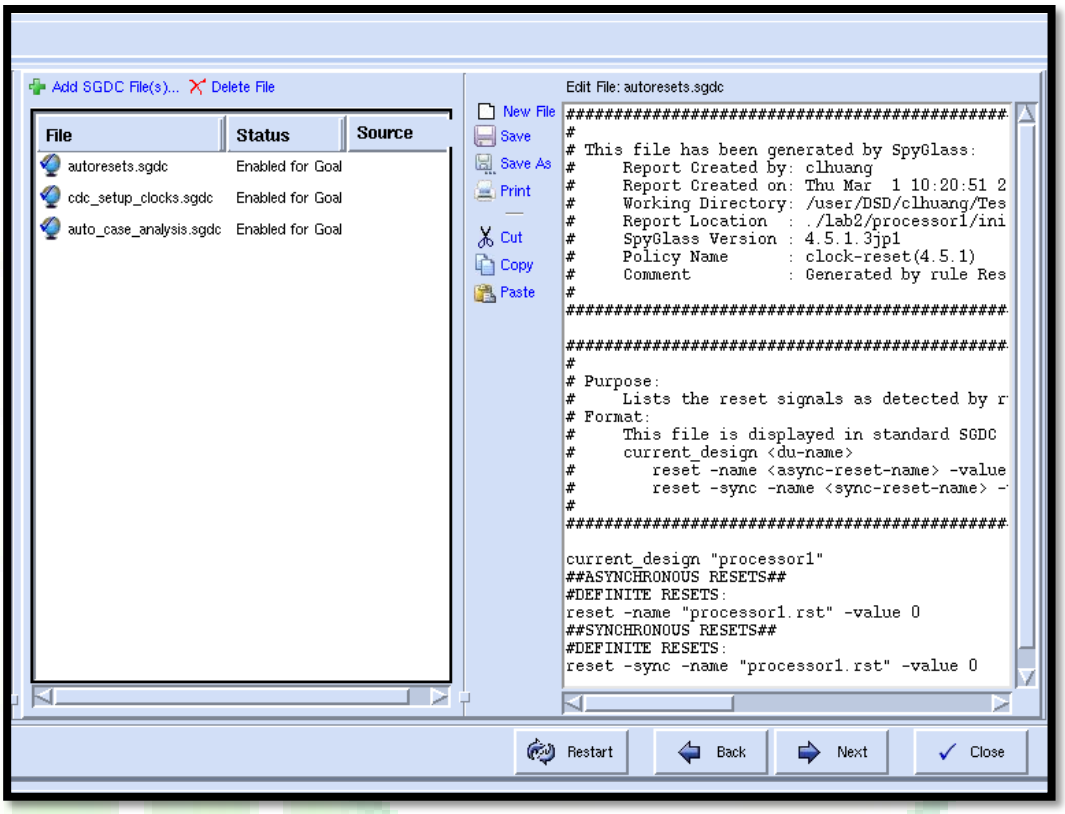
Do you have any SGDC files?	Yes
Do you want to import constraints from SDC files	No
Identify potential clocks used in the design	No


Do you have any SGDC files? [✓ Yes] No

Do you want to import constraints from SDC files? Yes [✓ No]

Identify potential clocks used in the design? Yes [✓ No]

11. Click "Next and check every SGDC file is setting correctly.

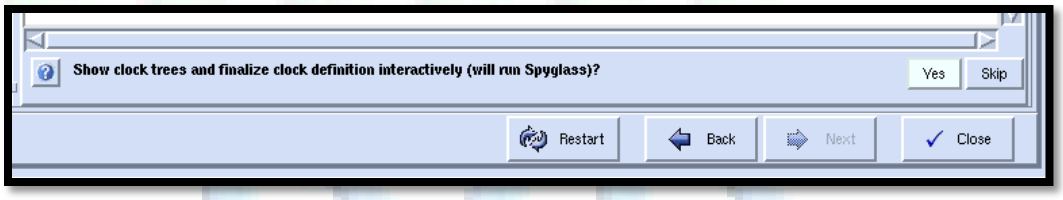


File	Status	Source
autoresets.sgdc	Enabled for Goal	
cdc_setup_clocks.sgdc	Enabled for Goal	
auto_case_analysis.sgdc	Enabled for Goal	

```
###############################################
#
# This file has been generated by SpyGlass:
#     Report Created by: clhuang
#     Report Created on: Thu Mar  1 10:20:51 2
#     Working Directory: /user/DSD/clhuang/Tes
#     Report Location  : ./lab2/processor1/ini
#     SpyGlass Version : 4.5.1.3jp1
#     Policy Name      : clock-reset(4.5.1)
#     Comment          : Generated by rule Res
#
###############################################

###############################################
#
# Purpose:
#     Lists the reset signals as detected by r
# Format:
#     This file is displayed in standard SGDC
#     current_design <du-name>
#        reset -name <async-reset-name> -value
#        reset -sync -name <sync-reset-name> -
#
###############################################

current_design "processor1"
##ASYNCHRONOUS RESETS##
#DEFINITE RESETS:
reset -name "processor1.rst" -value 0
##SYNCHRONOUS RESETS##
#DEFINITE RESETS:
reset -sync -name "processor1.rst" -value 0
```

12. Clock setup

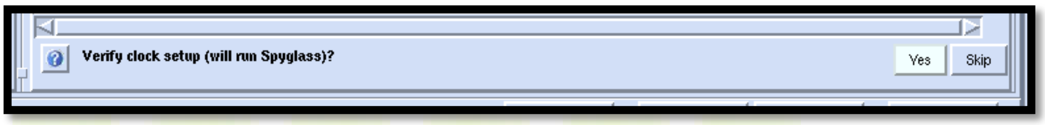
Show clock tree and finalize clock definition interactively?	**Yes**



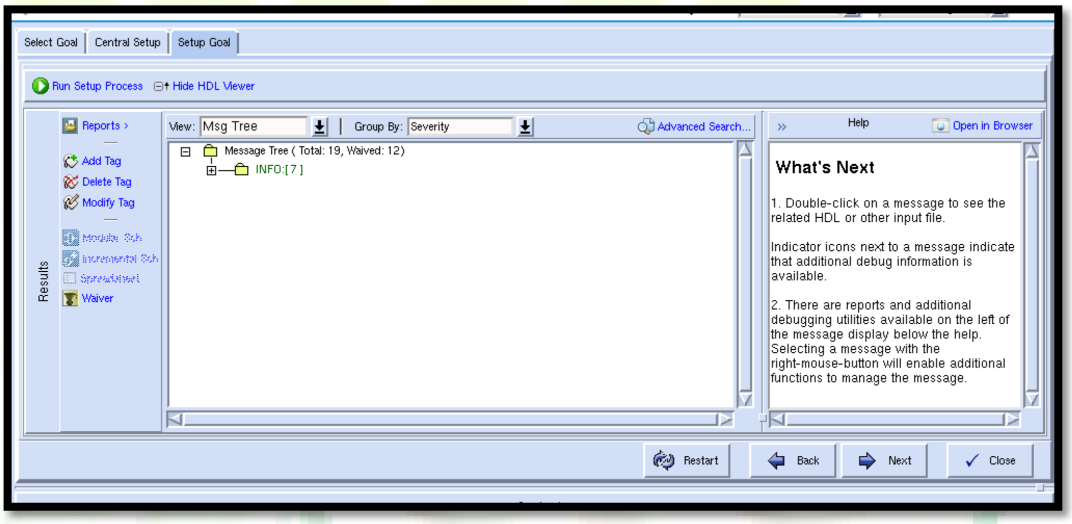
Show clock trees and finalize clock definition interactively (will run Spyglass)?

13. Click "Next" and perform verify clock setup.

Verify clock setup?	**Yes**



Verify clock setup (will run Spyglass)?



Message Tree (Total: 19, Waived: 12)
INFO:[7]

What's Next

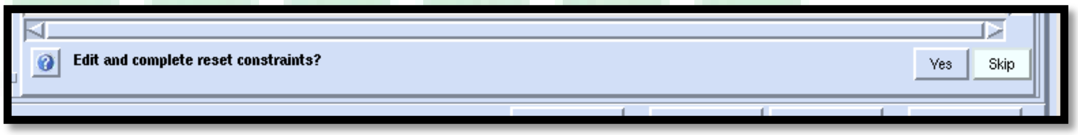
1. Double-click on a message to see the related HDL or other input file.

Indicator icons next to a message indicate that additional debug information is available.

2. There are reports and additional debugging utilities available on the left of the message display below the help. Selecting a message with the right-mouse-button will enable additional functions to manage the message.

14. Click "Next" and skip reset setup

Edit and complete reset constraints?	**Skip**



Edit and complete reset constraints?

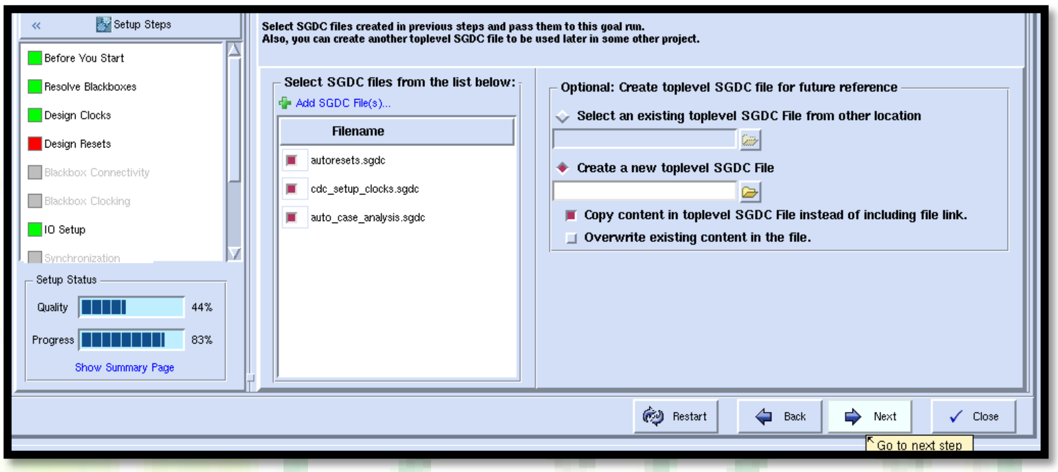
15. IO port setup

Edit and complete IO Domains?	**Yes**

Input	Inferred Clock(s)	Actual Clock
In2	processor1.clk2	**processor1.clk2**
Others	Default	Copy from inferred clocks column

16. Click “Next” and finish setup.

Copy content in toplevel SGDC file instead of including file link	**Check**
Others	Default



Select SGDC files created in previous steps and pass them to this goal run.
Also, you can create another toplevel SGDC file to be used later in some other project.

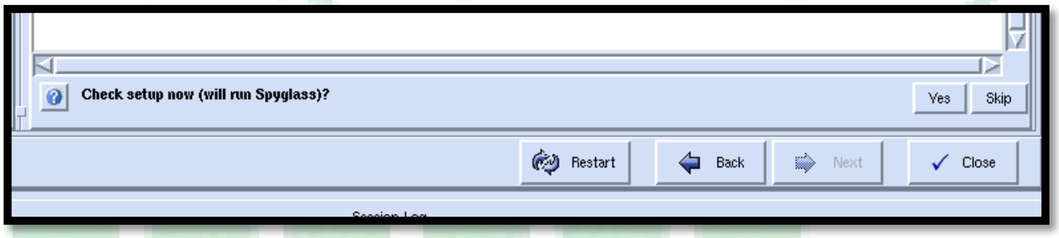
Select SGDC files from the list below:
Add SGDC File(s)...

Filename
autoresets.sgdc
cdc_setup_clocks.sgdc
auto_case_analysis.sgdc

Optional: Create toplevel SGDC file for future reference
Select an existing toplevel SGDC File from other location
Create a new toplevel SGDC File
Copy content in toplevel SGDC File instead of including file link.
Overwrite existing content in the file.

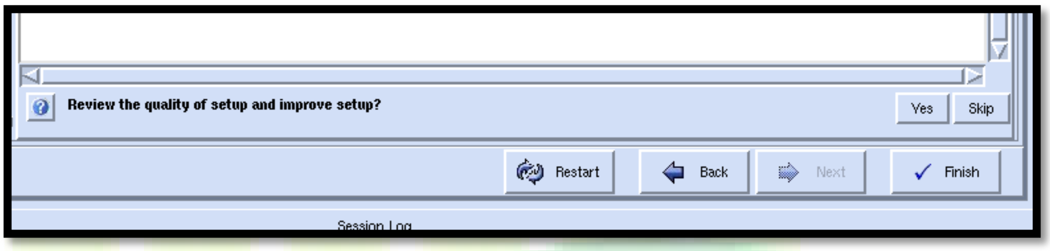
17. Click “Next” and check setup

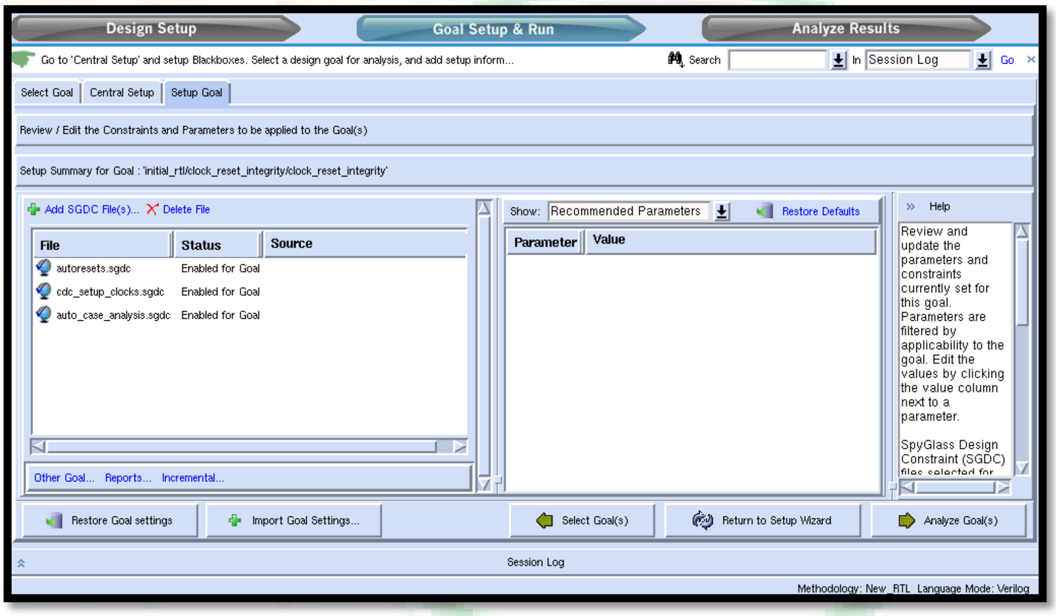
Check setup now	**Yes**



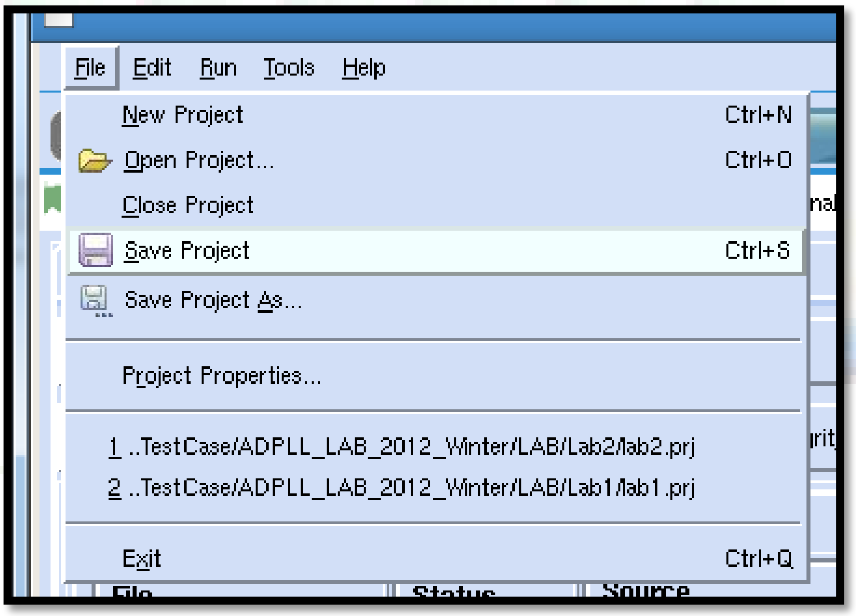
Check setup now (will run Spyglass)?

18. Click “Finish” and finish closure

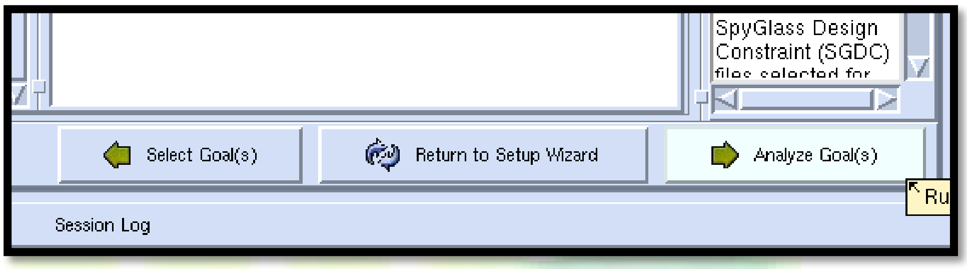



Design Setup
Goal Setup & Run
Analyze Results
Go to 'Central Setup' and setup Blackboxes. Select a design goal for analysis, and add setup inform...
Search
In
Session Log
Go
Select Goal
Central Setup
Setup Goal
Review / Edit the Constraints and Parameters to be applied to the Goal(s)
Setup Summary for Goal : 'initial_rtl/clock_reset_integrity/clock_reset_integrity'
Add SGDC File(s)...
Delete File
Show:
Recommended Parameters
Restore Defaults
Help
File
Status
Source
Parameter
Value
autoresets.sgdc Enabled for Goal
cdc_setup_clocks.sgdc Enabled for Goal
auto_case_analysis.sgdc Enabled for Goal
Review and update the parameters and constraints currently set for this goal. Parameters are filtered by applicability to the goal. Edit the values by clicking the value column next to a parameter.
SpyGlass Design Constraint (SGDC) files selected for
Other Goal...
Reports...
Incremental...
Restore Goal settings
Import Goal Settings...
Select Goal(s)
Return to Setup Wizard
Analyze Goal(s)
Session Log
Methodology: New_RTL Language Mode: Verilog

19. Save project

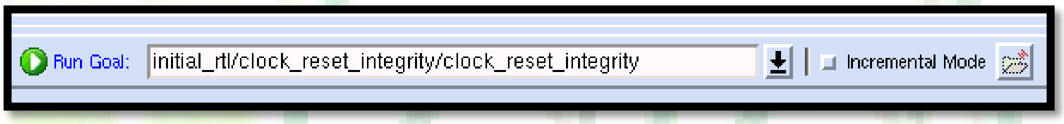


20. Analyze goal

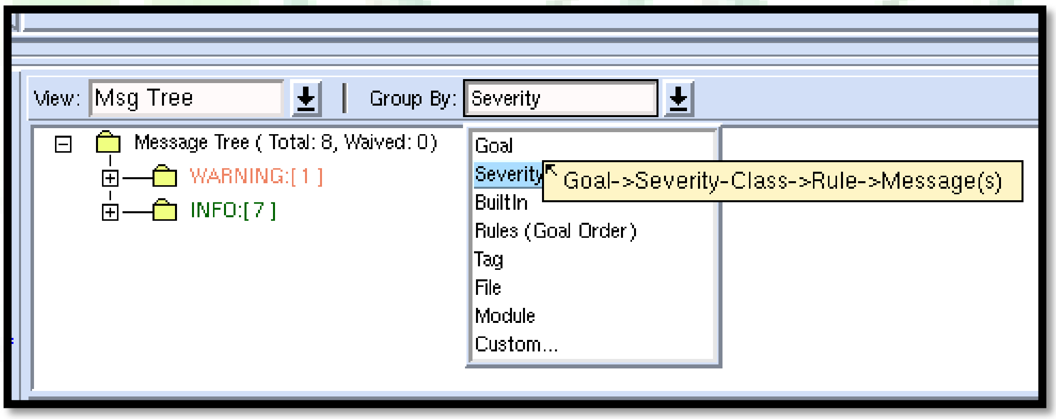


21. Click "Run goal" to run target goal

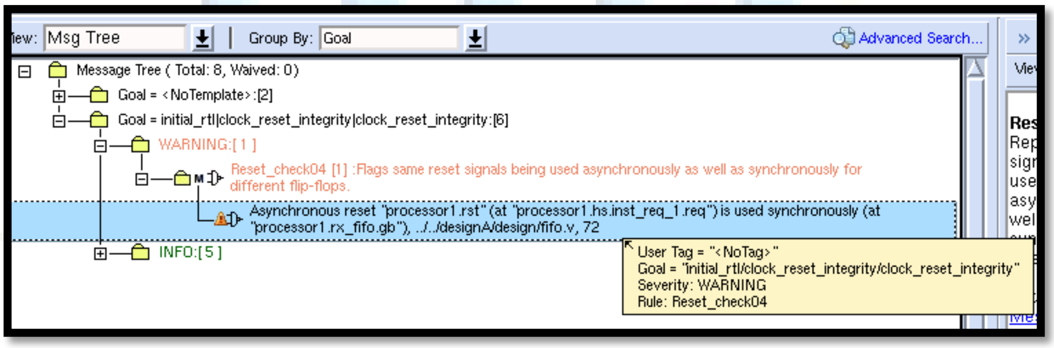
Run Goal	Initial_rtl/clock_reset_integrity/clock_reset_integrity



22. In the "Msg Tree" tab of the message window, group messages by "goal"

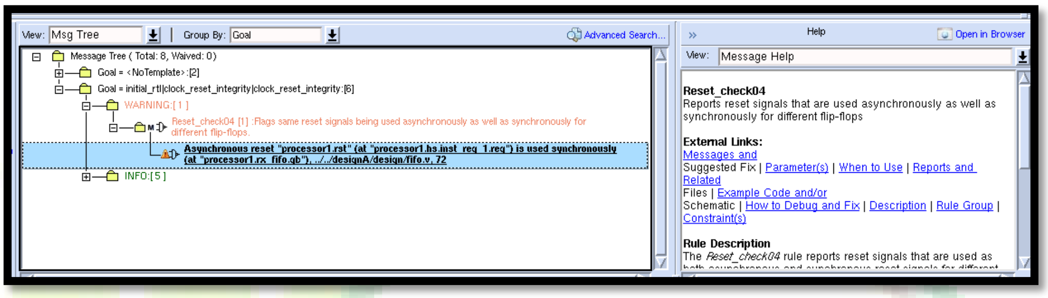


23. Expand the "clock_reset"integrity" goal folder and then expand the
"Reset_check04" rule folder.



```
Message Tree ( Total: 8, Waived: 0)
  Goal = <NoTemplate>:[2]
  Goal = initial_rtl|clock_reset_integrity|clock_reset_integrity:[6]
    WARNING:[ 1 ]
      Reset_check04 [1] :Flags same reset signals being used asynchronously as well as synchronously for
      different flip-flops.
        Asynchronous reset "processor1.rst" (at "processor1.hs.inst_req_1.req") is used synchronously (at
        "processor1.rx_fifo.gb"), ../../designA/design/fifo.v, 72
    INFO:[ 5 ]

User Tag = "<NoTag>"
Goal = "initial_rtl/clock_reset_integrity/clock_reset_integrity"
Severity: WARNING
Rule: Reset_check04
```

24. Double-click the “WARNING” message. The “Help” window tells us the reset signals that are used asynchronously as well as synchronously for different flip-flops.



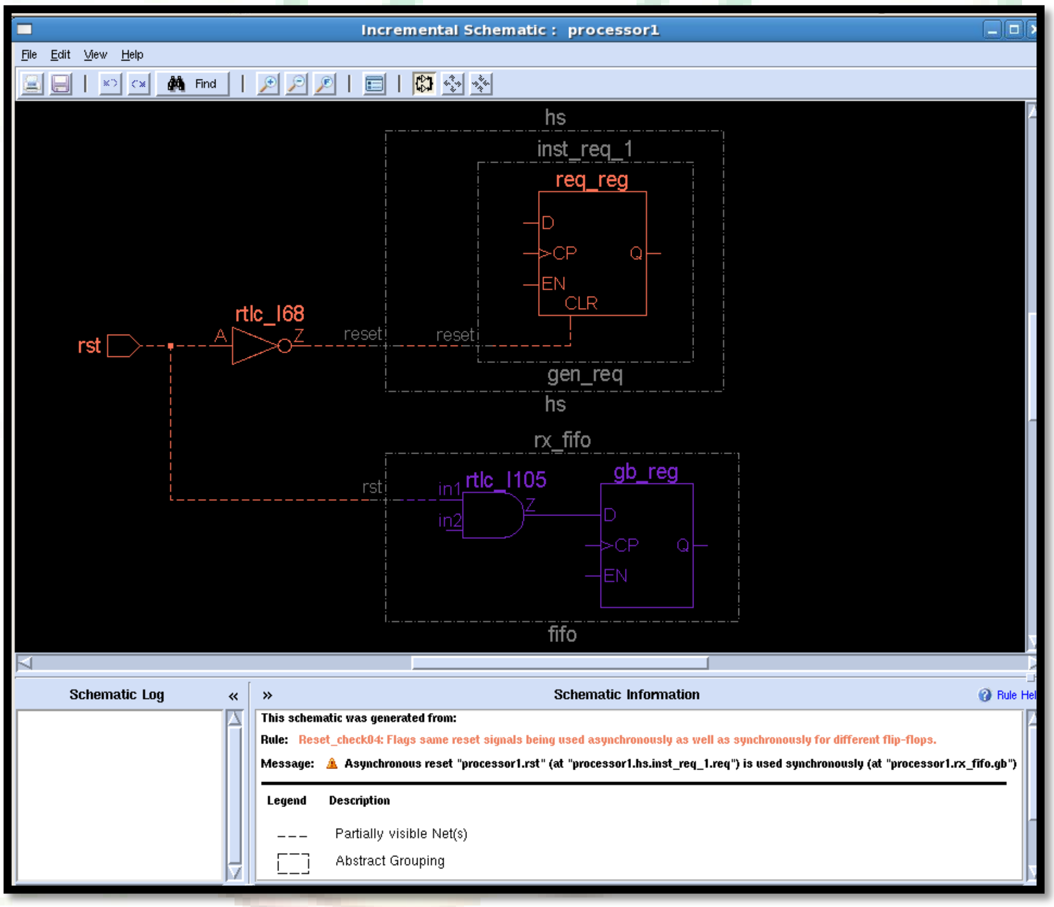
Message Tree (Total: 8, Waived: 0)
Goal = <NoTemplate>:[2]
Goal = initial_rtl|clock_reset_integrity|clock_reset_integrity:[6]
WARNING:[1]
Reset_check04 [1] :Flags same reset signals being used asynchronously as well as synchronously for different flip-flops.
Asynchronous reset "processor1.rst" (at "processor1.hs.inst_req_1.req") is used synchronously (at "processor1.rx_fifo.gb"), ../../designA/design/fifo.v, 72
INFO:[5]

Reset_check04
Reports reset signals that are used asynchronously as well as synchronously for different flip-flops

External Links:
Messages and Suggested Fix | Parameter(s) | When to Use | Reports and Related Files | Example Code and/or Schematic | How to Debug and Fix | Description | Rule Group | Constraint(s)

Rule Description
The *Reset_check04* rule reports reset signals that are used as

25. Invoke incremental schematic by pressing hotkey **<i>** and you will see 2 flip-flop fed by reset signal in different modes.



This schematic was generated from:

Rule: Reset_check04: Flags same reset signals being used asynchronously as well as synchronously for different flip-flops.

Message: Asynchronous reset "processor1.rst" (at "processor1.hs.inst_req_1.req") is used synchronously (at "processor1.rx_fifo.gb")

Legend	Description
- - -	Partially visible Net(s)
⬚	Abstract Grouping

[Tips]

If it warning message is being don intentionally in design then it’s OK and a waiver should be added for this violation, else, it is recommended to decide on a uniform reset strategy and stick to it.

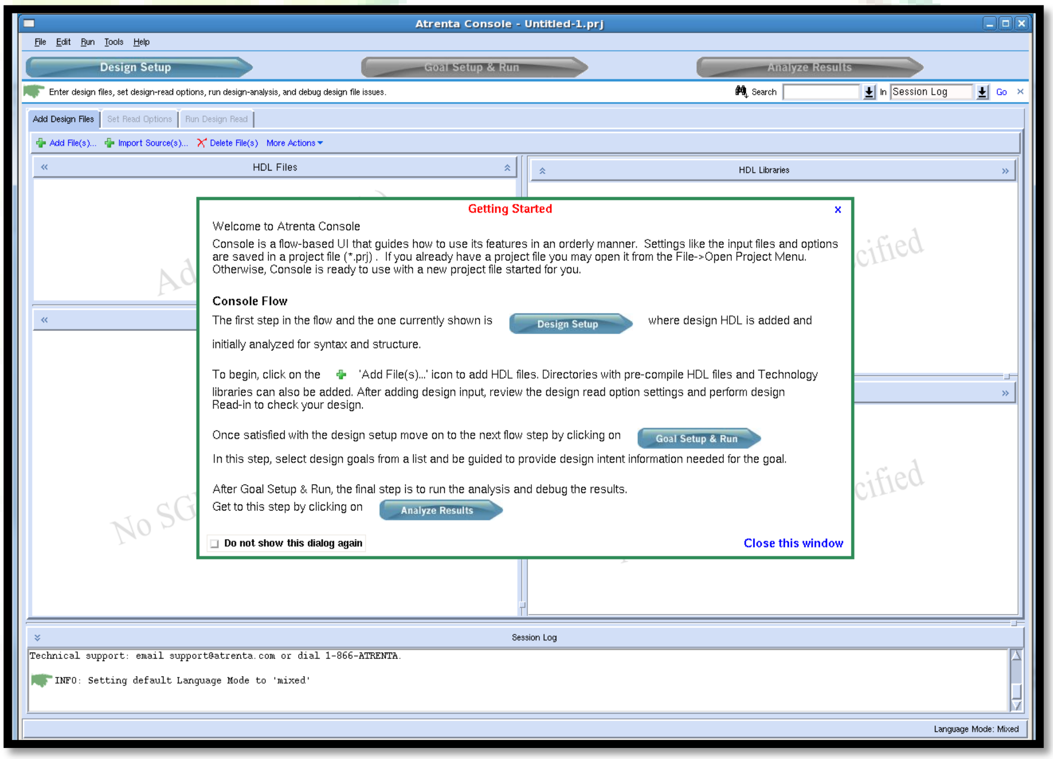
Lab 7 (SKIP) Performing metastability checks

1. Change directory to lab 7

   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_7

2. Open the SpyGlass in interactive mode

   Unix% spyglass



Enter design files, set design-read options, run design-analysis, and debug design file issues.

Getting Started

Welcome to Atrenta Console

Console is a flow-based UI that guides how to use its features in an orderly manner. Settings like the input files and options are saved in a project file (*.prj) . If you already have a project file you may open it from the File->Open Project Menu. Otherwise, Console is ready to use with a new project file started for you.

Console Flow

The first step in the flow and the one currently shown is Design Setup where design HDL is added and initially analyzed for syntax and structure.

To begin, click on the 'Add File(s)...' icon to add HDL files. Directories with pre-compile HDL files and Technology libraries can also be added. After adding design input, review the design read option settings and perform design Read-in to check your design.

Once satisfied with the design setup move on to the next flow step by clicking on Goal Setup & Run

In this step, select design goals from a list and be guided to provide design intent information needed for the goal.

After Goal Setup & Run, the final step is to run the analysis and debug the results.
Get to this step by clicking on Analyze Results

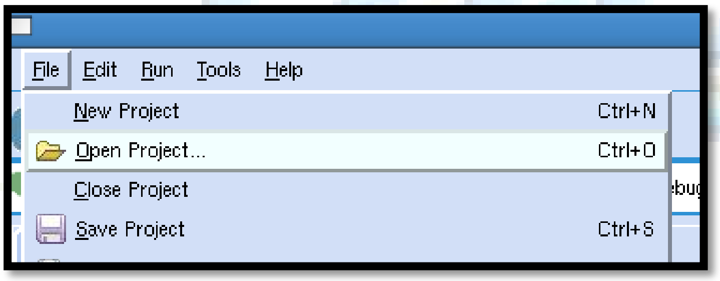
Do not show this dialog again

Close this window

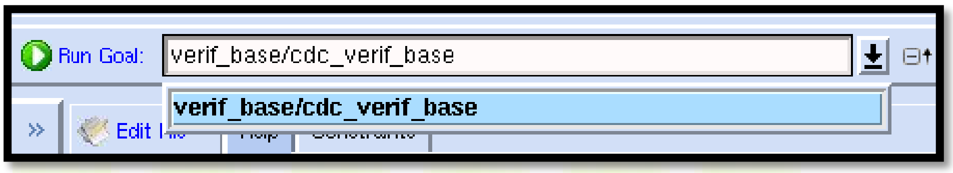
```
Technical support: email support@atrenta.com or dial 1-866-ATRENTA.
INFO: Setting default Language Mode to 'mixed'
```

3. Open project

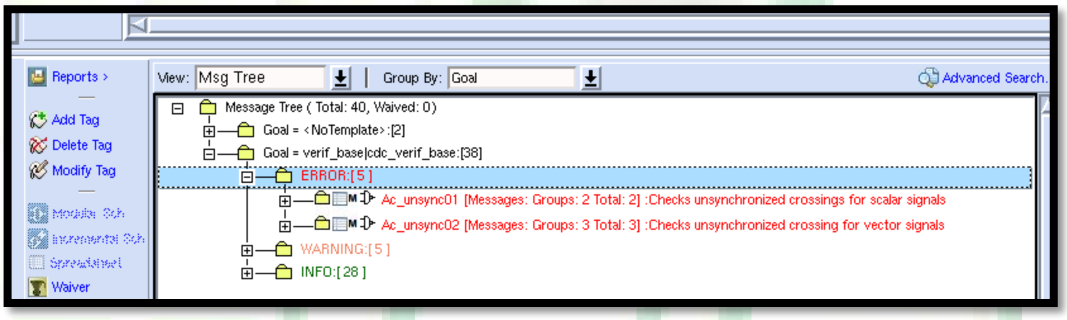
Project file name	Lab7.prj



4. Switch run goal to “verif_base/cdc_verif_base”

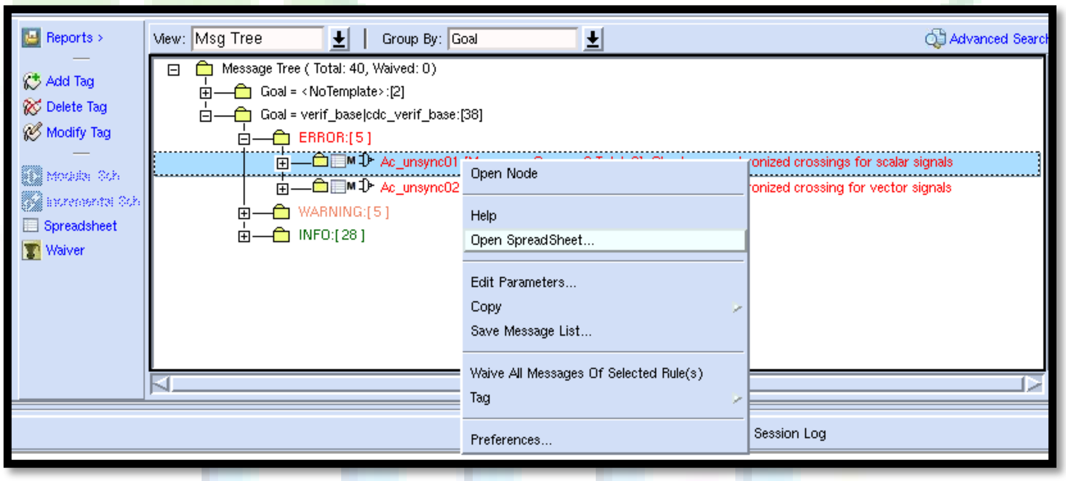


5. In the message tree view, double-click the ERROR folder.

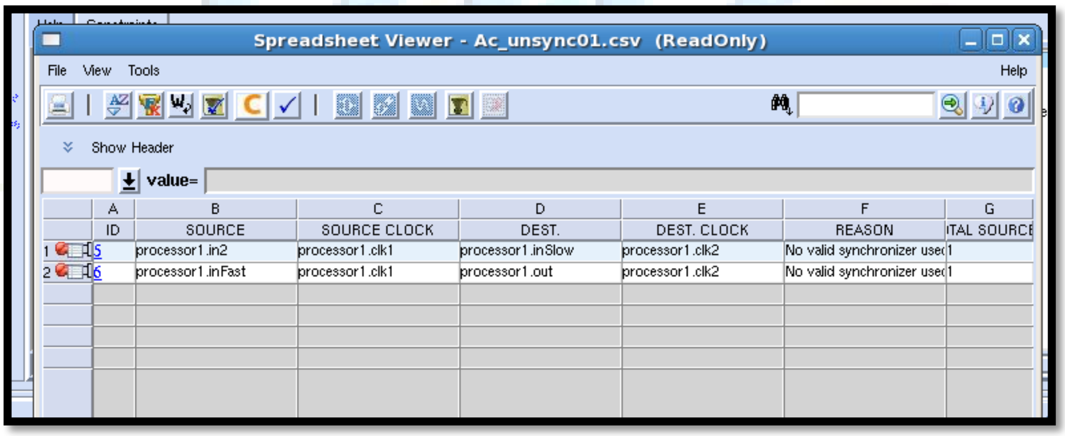


```
Message Tree ( Total: 40, Waived: 0)
  Goal = <NoTemplate>:[2]
  Goal = verif_base|cdc_verif_base:[38]
    ERROR:[5]
      Ac_unsync01 [Messages: Groups: 2 Total: 2] :Checks unsynchronized crossings for scalar signals
      Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals
    WARNING:[5]
    INFO:[28]
```

6. Select the “Ac_unsync01” message and right-click. In the context window, select the “Open Spreadsheet…” option. And then, you will see the spreadsheet viewer.



```
Message Tree ( Total: 40, Waived: 0)
  Goal = <NoTemplate>:[2]
  Goal = verif_base|cdc_verif_base:[38]
    ERROR:[5]
      Ac_unsync01 ...onized crossings for scalar signals
      Ac_unsync02 ...onized crossing for vector signals
    WARNING:[5]
    INFO:[28]
```

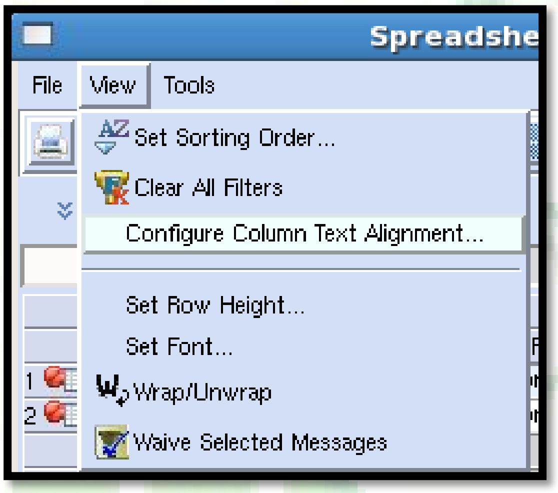


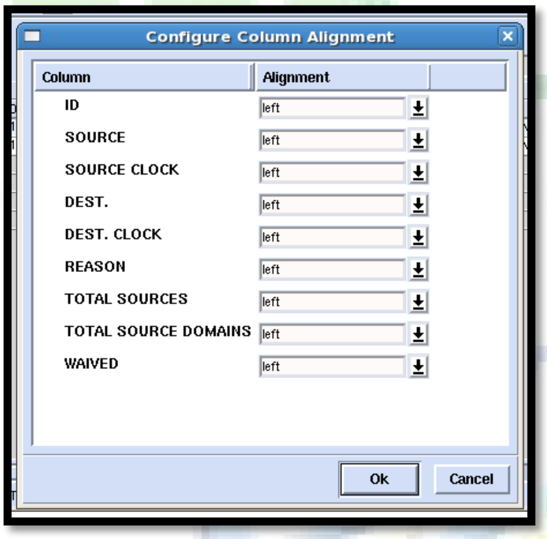
	A	B	C	D	E	F	G
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON	ITAL SOURCE
1	5	processor1.in2	processor1.clk1	processor1.inSlow	processor1.clk2	No valid synchronizer used	1
2	6	processor1.inFast	processor1.clk1	processor1.out	processor1.clk2	No valid synchronizer used	1

[Tips]

Click any column will sort that column, for example you may like to sort source or destination in a real design scenario.

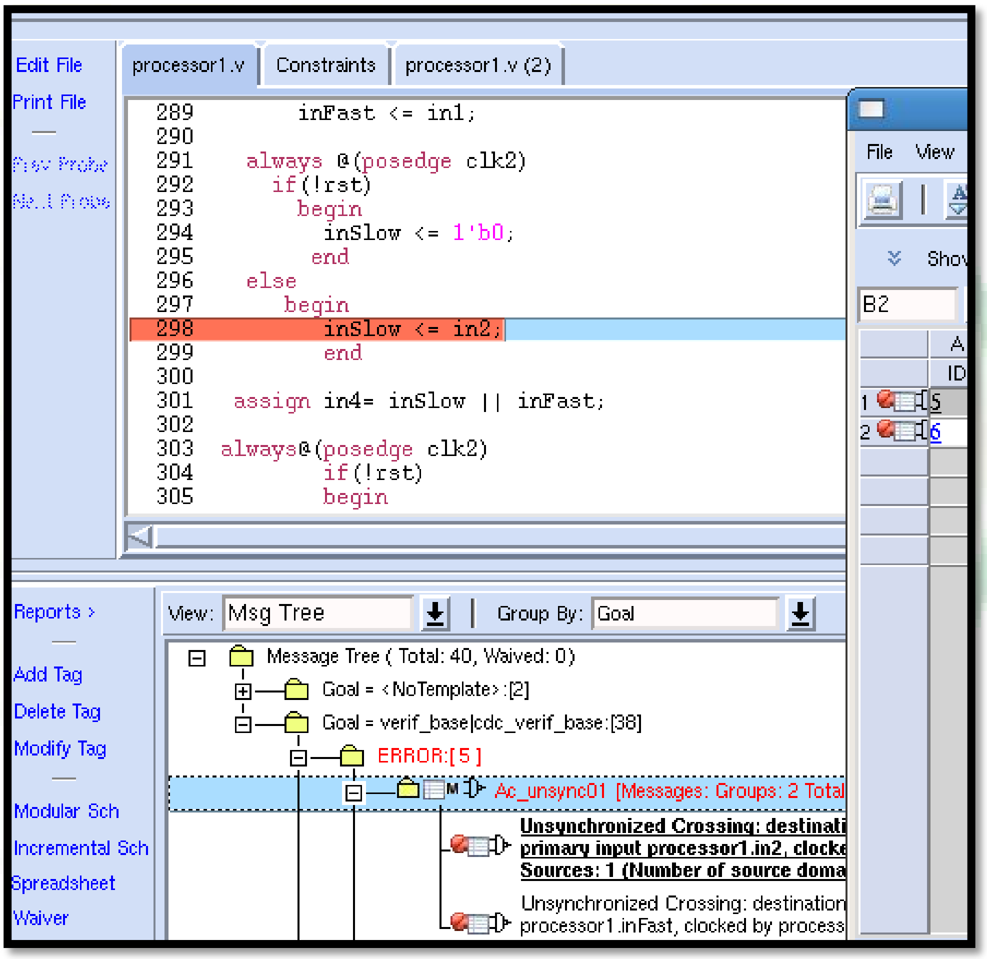
7. Since hierarchical signal names are often longer and may not be visible in spreadsheet cell, default the name are LEFT justified, but you can change it. View => Configure Column Text Alignment”




Configure Column Alignment

Column	Alignment
ID	left
SOURCE	left
SOURCE CLOCK	left
DEST.	left
DEST. CLOCK	left
REASON	left
TOTAL SOURCES	left
TOTAL SOURCE DOMAINS	left
WAIVED	left

8. Let’s scroll over to the left of the viewer, in the first column, there is an ID field, click the first entry in this field. Notice that the corresponding message in the message tree view of the CONSOLE as well as the line of code in the RTL view is being high-lighted.



```
        inFast <= in1;

   always @(posedge clk2)
     if(!rst)
       begin
         inSlow <= 1'b0;
       end
   else
      begin
         inSlow <= in2;
         end

  assign in4= inSlow || inFast;

 always@(posedge clk2)
         if(!rst)
         begin
```

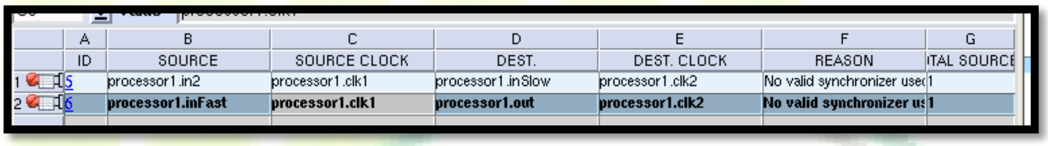
		ID
1		5
2		6

```
Message Tree ( Total: 40, Waived: 0)
  Goal = <NoTemplate>:[2]
  Goal = verif_base|cdc_verif_base:[38]
    ERROR:[5]
      Ac_unsync01 [Messages: Groups: 2 Total
        Unsynchronized Crossing: destinati
        primary input processor1.in2, clocke
        Sources: 1 (Number of source doma
        Unsynchronized Crossing: destination
        processor1.inFast, clocked by process
```

9. You can use spreadsheet viewer to interactively to apply cdc_false_path on a crossing

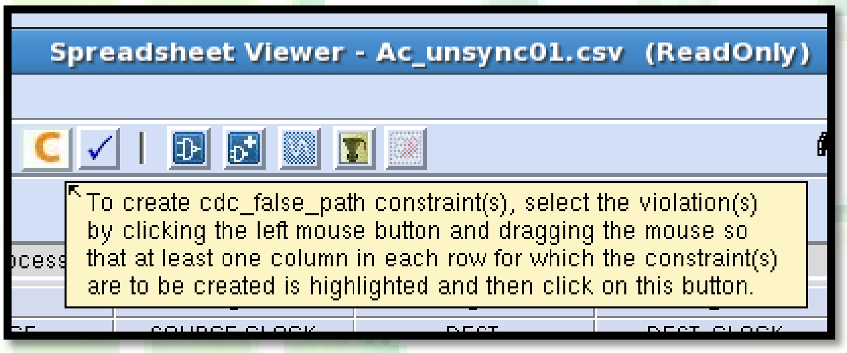
Click the cell to select the particular path

SOUCE	Processor1.InFast



	A	B	C	D	E	F	G
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON	ITAL SOURCE
1	5	processor1.in2	processor1.clk1	processor1.inSlow	processor1.clk2	No valid synchronizer used	1
2	6	**processor1.inFast**	**processor1.clk1**	**processor1.out**	**processor1.clk2**	**No valid synchronizer us**	**1**

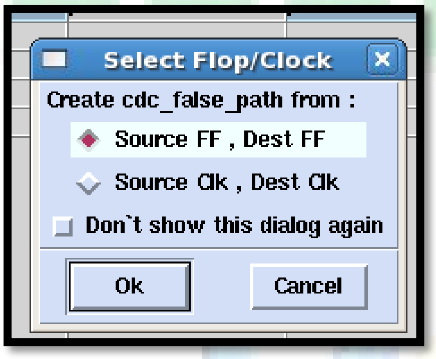
Click the “C” button to set cdc_false_path



Spreadsheet Viewer - Ac_unsync01.csv (ReadOnly)

To create cdc_false_path constraint(s), select the violation(s) by clicking the left mouse button and dragging the mouse so that at least one column in each row for which the constraint(s) are to be created is highlighted and then click on this button.

Select “Source FF, Dest FF” and click “OK”



Select Flop/Clock

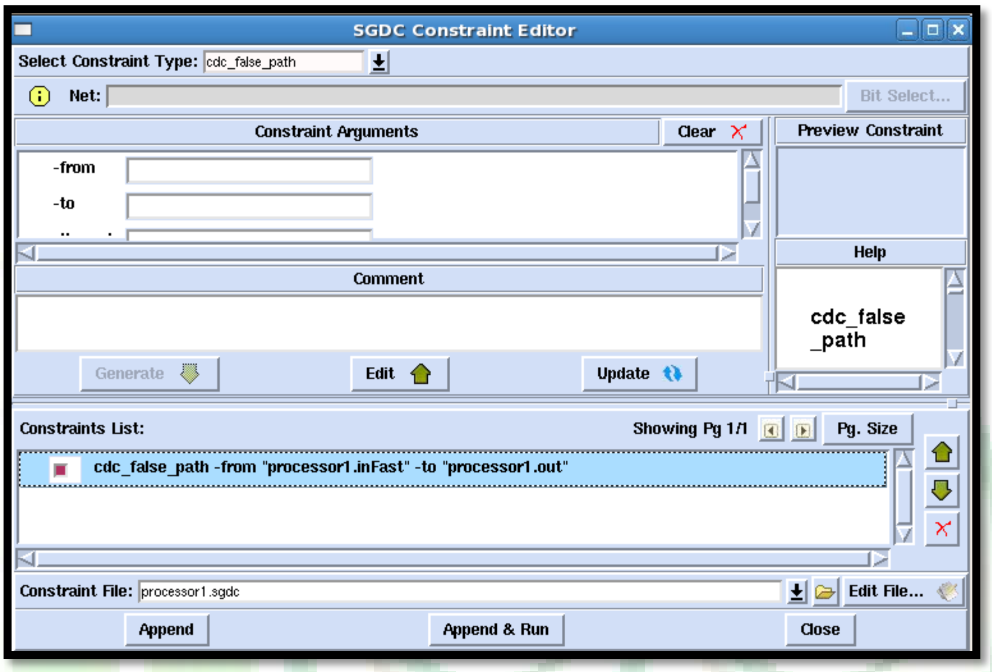
Create cdc_false_path from :

[✓ Source FF , Dest FF]

Source Clk , Dest Clk

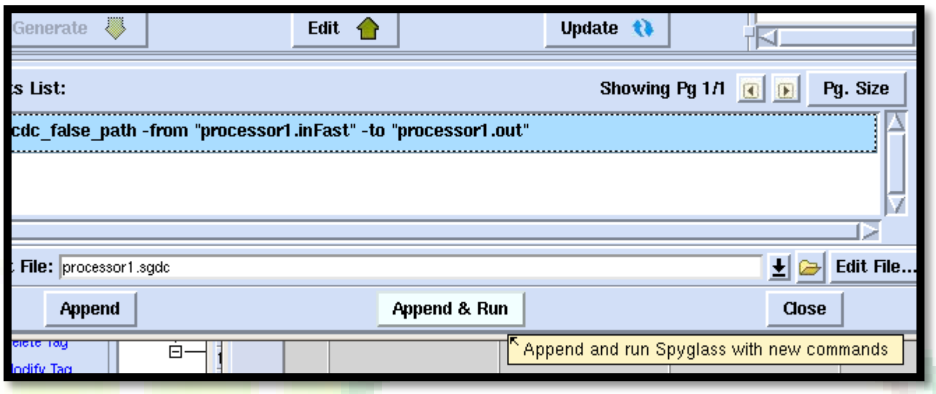
☐ Don`t show this dialog again

cdc_false_path will be generated in constraints editor



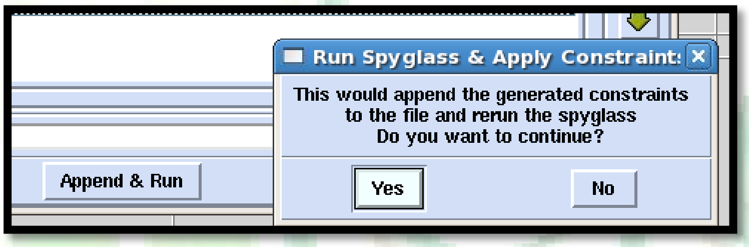
```
cdc_false_path -from "processor1.inFast" -to "processor1.out"
```

Finally click “Append & Run” to update the constraints file and run the analysis.

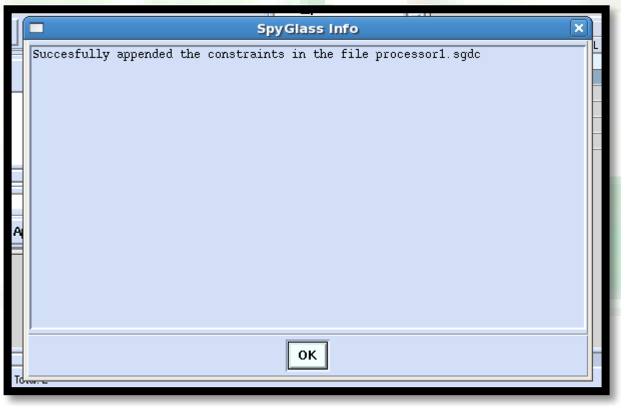


```
cdc_false_path -from "processor1.inFast" -to "processor1.out"
```

File: processor1.sgdc



This would append the generated constraints to the file and rerun the spyglass
Do you want to continue?

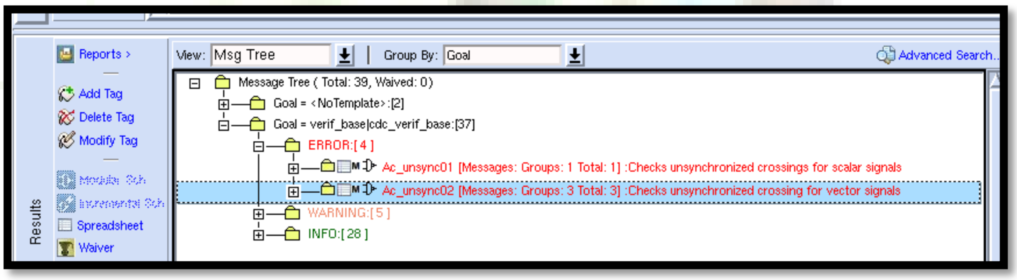


```
Succesfully appended the constraints in the file processor1.sgdc
```

[Tips]

If you have more paths to be set cdc_false_path on, repeat the above steps.

10. Open the “Ac_unsync02” rule folder.

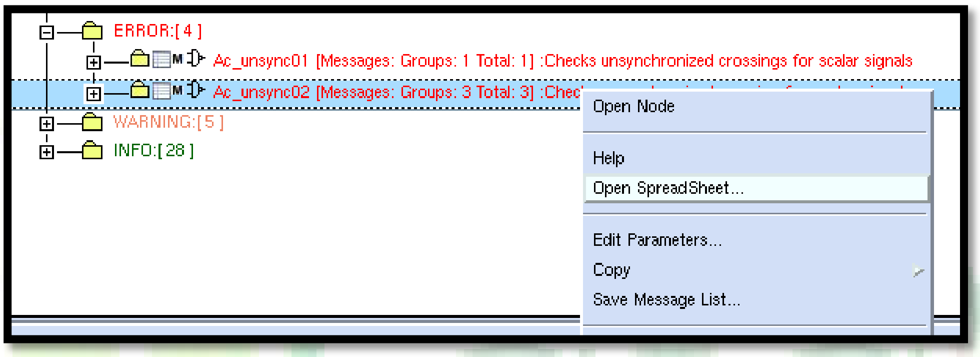


```
Message Tree ( Total: 39, Waived: 0)
  Goal = <NoTemplate>:[2]
  Goal = verif_base|cdc_verif_base:[37]
    ERROR:[ 4 ]
      Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
      Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals
    WARNING:[ 5 ]
    INFO:[ 28 ]
```

[Think about it]

What’s the “Ac_unsync02” rule?

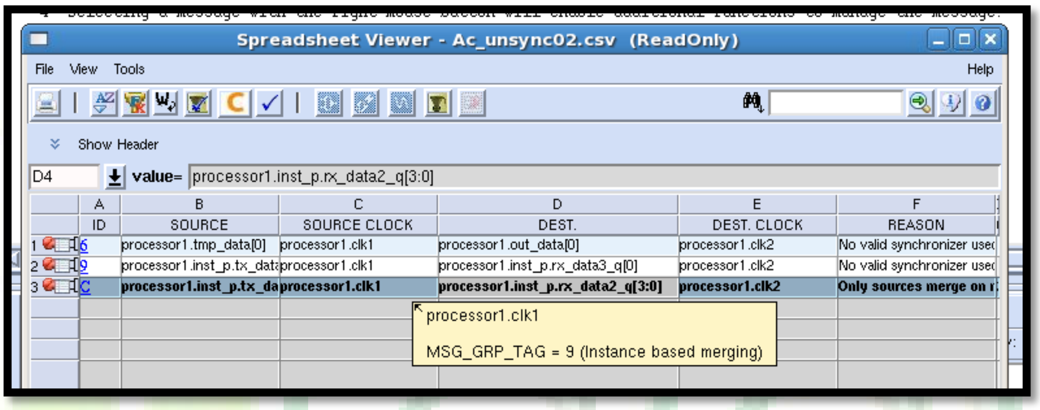
11. Right-click on “Ac_unsync02” folder and select “Open Spreadsheet”



ERROR:[4]
Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
Ac_unsync02 [Messages: Groups: 3 Total: 3] :Chec
WARNING:[5]
INFO:[28]

12. In the “DEST” column locate “top.processor1.inst_p.rx_data2_q[3:0]” and click on “ID” to open violation spreadsheet and at the same time it will also cross-probe to Msg tree.

DEST.	top.processor1.inst_p.rx_data2_q[3:0]

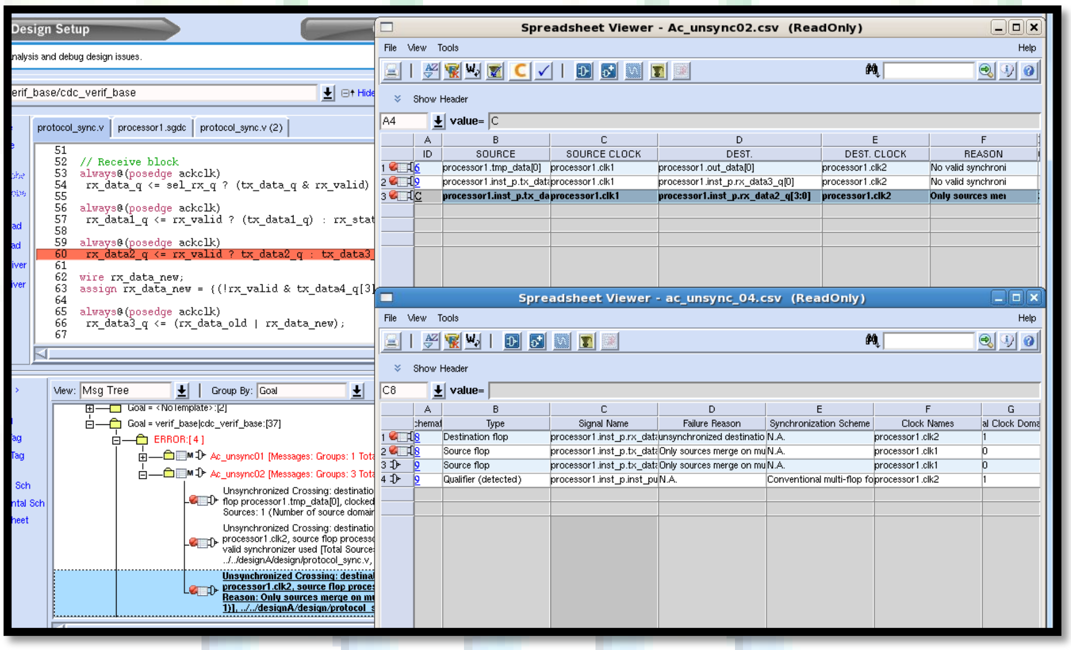


D4 value= processor1.inst_p.rx_data2_q[3:0]

	A	B	C	D	E	F
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON
1	6	processor1.tmp_data[0]	processor1.clk1	processor1.out_data[0]	processor1.clk2	No valid synchronizer usec
2	9	processor1.inst_p.tx_data	processor1.clk1	processor1.inst_p.rx_data3_q[0]	processor1.clk2	No valid synchronizer usec
3	C	**processor1.inst_p.tx_da**	**processor1.clk1**	**processor1.inst_p.rx_data2_q[3:0]**	**processor1.clk2**	**Only sources merge on r**

processor1.clk1

MSG_GRP_TAG = 9 (Instance based merging)



```
// Receive block
always@(posedge ackclk)
 rx_data_q <= sel_rx_q ? (tx_data_q & rx_valid)

always@(posedge ackclk)
 rx_data1_q <= rx_valid ? (tx_data1_q) : rx_stat

always@(posedge ackclk)
 rx_data2_q <= rx_valid ? tx_data2_q : tx_data3_

wire rx_data_new;
assign rx_data_new = {(!rx_valid & tx_data4_q[3]

always@(posedge ackclk)
 rx_data3_q <= (rx_data_old | rx_data_new);
```

View: Msg Tree | Group By: Goal

```
Goal = <NoTemplate>:[2]
Goal = verif_base|cdc_verif_base:[37]
  ERROR:[ 4 ]
    Ac_unsync01 [Messages: Groups: 1 Tota
    Ac_unsync02 [Messages: Groups: 3 Tota
      Unsynchronized Crossing: destinatio
      flop processor1.tmp_data[0], clocked
      Sources: 1 (Number of source domair
      Unsynchronized Crossing: destinatio
      processor1.clk2, source flop processo
      valid synchronizer used [Total Source
      ../../designA/design/protocol_sync.v,
      Unsynchronized Crossing: destinat
      processor1.clk2, source flop proce
      Reason: Only sources merge on mu
      1)], ../../designA/design/protocol_s
```

Spreadsheet Viewer - Ac_unsync02.csv (ReadOnly)

A4 value= C

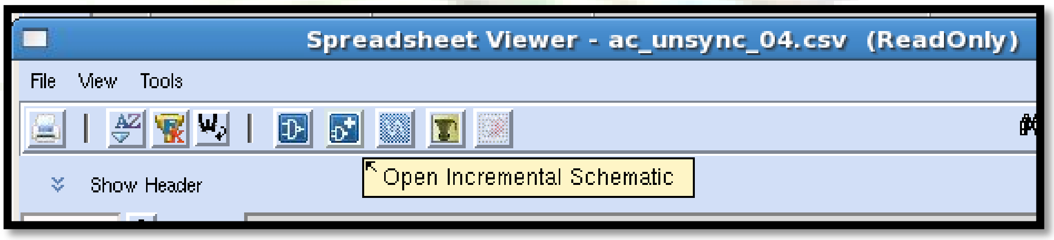
	A	B	C	D	E	F
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON
1	6	processor1.tmp_data[0]	processor1.clk1	processor1.out_data[0]	processor1.clk2	No valid synchroni
2	9	processor1.inst_p.tx_data	processor1.clk1	processor1.inst_p.rx_data3_q[0]	processor1.clk2	No valid synchroni
3	C	**processor1.inst_p.tx_da**	**processor1.clk1**	**processor1.inst_p.rx_data2_q[3:0]**	**processor1.clk2**	**Only sources mei**

Spreadsheet Viewer - ac_unsync_04.csv (ReadOnly)

C8 value=

	A	B	C	D	E	F	G
	chemat	Type	Signal Name	Failure Reason	Synchronization Scheme	Clock Names	al Clock Doma
1	8	Destination flop	processor1.inst_p.rx_data	unsynchronized destinatio	N.A.	processor1.clk2	1
2	8	Source flop	processor1.inst_p.tx_data	Only sources merge on mu	N.A.	processor1.clk1	0
3	9	Source flop	processor1.inst_p.tx_data	Only sources merge on mu	N.A.	processor1.clk1	0
4	9	Qualifier (detected)	processor1.inst_p.inst_pu	N.A.	Conventional multi-flop fo	processor1.clk2	1

13. At the spreadsheet viewer “ac_unsync04.asv (ReadOnly), click “incremental Sch” icon to debug.


Spreadsheet Viewer - ac_unsync_04.csv (ReadOnly)
File View Tools
Show Header
Open Incremental Schematic

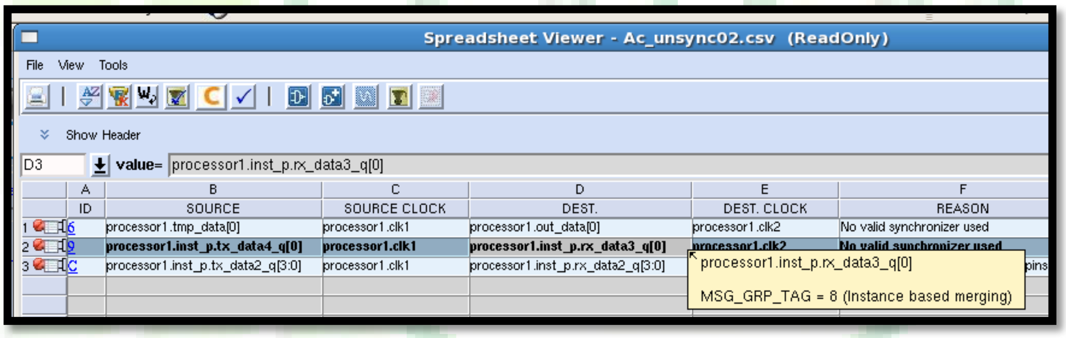
[Think about it]

The failure reason it seems that MUX inputs are fed by asynchronous source. But why “rx_data2_q” is reported as unsynchronized?
(HINT: Double clock the MUX input!)

[Tips]

SpyGlass, by default, reports all such crossing as unsynchronized/metastable. Apply a “cdc_false_path” constraint on paths that are having exception such as static or test or debug logic.

14. In the spreadsheet for “Ac_unsync02” rule violation locate destination “top.processor1.inst_p.rx_data3_q[3:0]” in “DEST” column



Spreadsheet Viewer - Ac_unsync02.csv (ReadOnly)

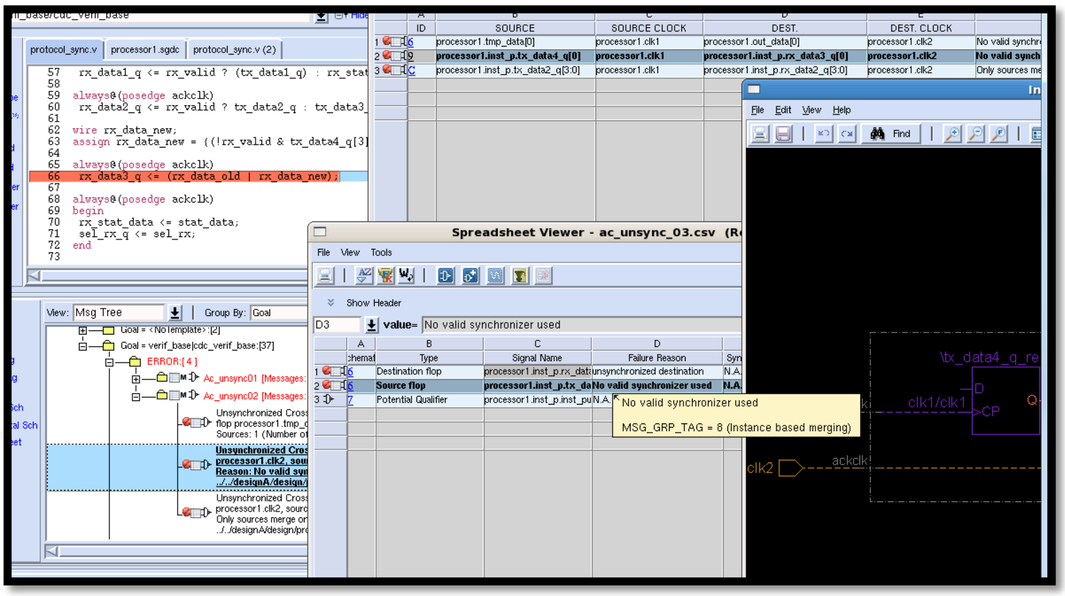
D3 value= processor1.inst_p.rx_data3_q[0]

	A	B	C	D	E	F
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON
1	6	processor1.tmp_data[0]	processor1.clk1	processor1.out_data[0]	processor1.clk2	No valid synchronizer used
2	9	**processor1.inst_p.tx_data4_q[0]**	**processor1.clk1**	**processor1.inst_p.rx_data3_q[0]**	**processor1.clk2**	**No valid synchronizer used**
3	C	processor1.inst_p.tx_data2_q[3:0]	processor1.clk1	processor1.inst_p.rx_data2_q[3:0]		pins

processor1.inst_p.rx_data3_q[0]

MSG_GRP_TAG = 8 (Instance based merging)

15. Click on the “ID” field and you will see this will cross probe the violation in schematic/Msg Tree, also a violation spreadsheet will open.



```
rx_data1_q <= rx_valid ? (tx_data1_q) : rx_stat
always@(posedge ackclk)
rx_data2_q <= rx_valid ? tx_data2_q : tx_data3_
wire rx_data_new;
assign rx_data_new = {(!rx_valid & tx_data4_q[3]
always@(posedge ackclk)
rx_data3_q <= (rx_data_old | rx_data_new);
always@(posedge ackclk)
begin
rx_stat_data <= stat_data;
sel_rx_q <= sel_rx;
end
```

Spreadsheet Viewer - ac_unsync_03.csv

D3 value= No valid synchronizer used

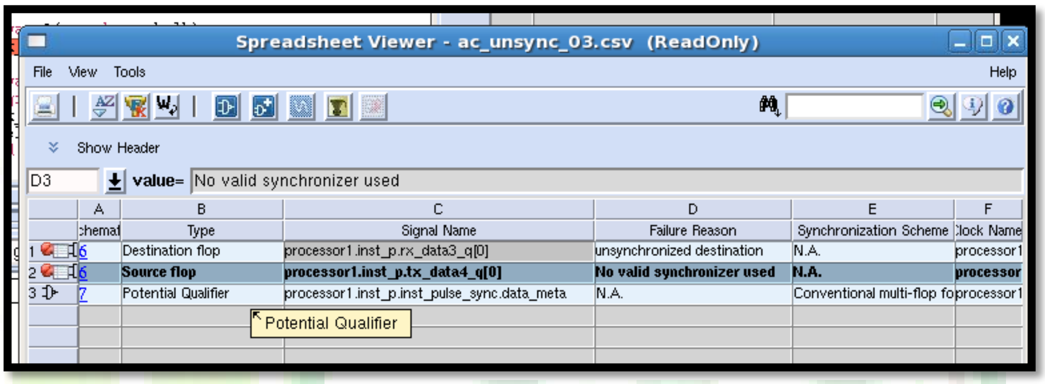
	A	B	C	D	
	chemat	Type	Signal Name	Failure Reason	Syn
1	6	Destination flop	processor1.inst_p.rx_dat	unsynchronized destination	N.A.
2	6	**Source flop**	**processor1.inst_p.tx_da**	**No valid synchronizer used**	**N.A.**
3	7	Potential Qualifier	processor1.inst_p.inst_pu	N.A.	

No valid synchronizer used

MSG_GRP_TAG = 8 (Instance based merging)

protocol_sync.v
processor1.sgdc
protocol_sync.v (2)
View: Msg Tree
Group By: Goal
Goal = <No Template>:[2]
Goal = verif_base|cdc_verif_base:[37]
ERROR:[4]
Ac_unsync01 [Messages:
Ac_unsync02 [Messages:
Unsynchronized Cross flop processor1.tmp_ Sources: 1 (Number of
Unsynchronized Cros processor1.clk2, sou Reason: No valid syn ../../designA/design/
Unsynchronized Cross processor1.clk2, sourc Only sources merge or ../../designA/design/pr
\tx_data4_q_re
clk1/clk1
D
CP
Q
clk2
ackclk

16. As you can see above reason of failure is “No valid synchronizer” used, which means tool is unable to find any synchronizer or synchronized control line. Also tool is also showing a potential qualifier for this crossing.



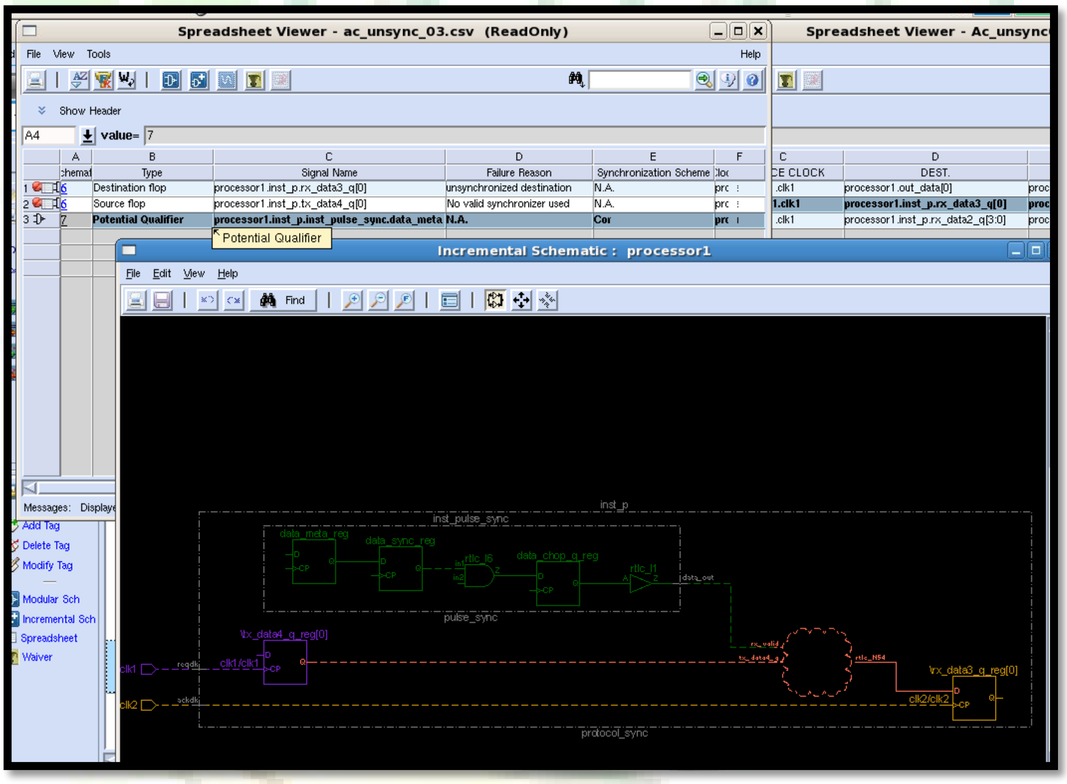
Spreadsheet Viewer - ac_unsync_03.csv (ReadOnly)

D3 value= No valid synchronizer used

	A	B	C	D	E	F
	chemat	Type	Signal Name	Failure Reason	Synchronization Scheme	Clock Name
1	6	Destination flop	processor1.inst_p.rx_data3_q[0]	unsynchronized destination	N.A.	processor1
2	6	**Source flop**	**processor1.inst_p.tx_data4_q[0]**	**No valid synchronizer used**	**N.A.**	**processor**
3	7	Potential Qualifier	processor1.inst_p.inst_pulse_sync.data_meta	N.A.	Conventional multi-flop fo	processor1

Potential Qualifier

17. Locate the “Potential Qualifier” at “Type” column and click on the “ID” field to cross-probe to schematic and open “Incremental Sch” window.



Spreadsheet Viewer - ac_unsync_03.csv (ReadOnly)

A4 value= 7

	A	B	C	D	E	F
	chemat	Type	Signal Name	Failure Reason	Synchronization Scheme	Cloc
1	6	Destination flop	processor1.inst_p.rx_data3_q[0]	unsynchronized destination	N.A.	prc
2	6	Source flop	processor1.inst_p.tx_data4_q[0]	No valid synchronizer used	N.A.	prc
3	7	**Potential Qualifier**	**processor1.inst_p.inst_pulse_sync.data_meta**	**N.A.**	**Cor**	**prc**

Spreadsheet Viewer - Ac_unsync

	C	D
	CE CLOCK	DEST.
1	.clk1	processor1.out_data[0]
2	**1.clk1**	**processor1.inst_p.rx_data3_q[0]**
3	.clk1	processor1.inst_p.rx_data2_q[3:0]

Potential Qualifier

Incremental Schematic : processor1

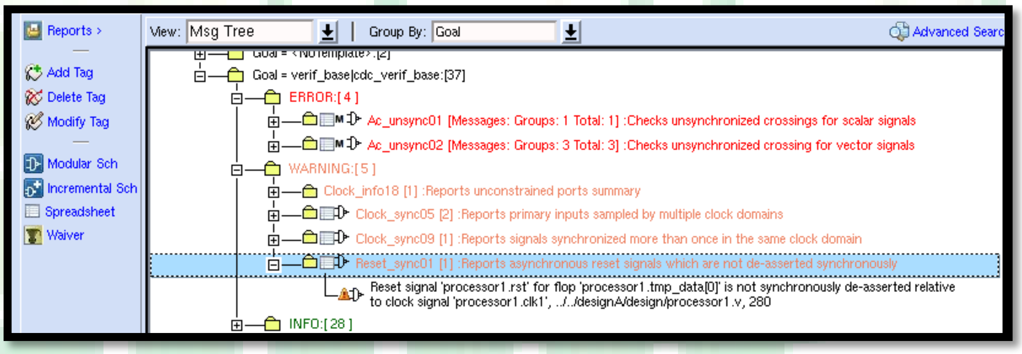
[Think about it]

From the schematic it seems qualifier is a double flip-flop synchronized signal and gating the source on AND gate. There is potential qualifier shown for “rx_data3”. If as a designer you accept this potential qualifier then what is the risk you are taking?

(HINT: Trace fan-in of OR gate which comes in between crossing, no asynchronous source should feed into OR gate other input)

18. Close all the spreadsheet viewer and incremental schematic windows.

19. To check the reset synchronization, double-click the “Reset_sync01” rule folder (under the Warning folder). You will see violation reported for the reset signal “rst”.



```
Goal = <NoTemplate>:[2]
Goal = verif_base|cdc_verif_base:[37]
    ERROR:[ 4 ]
        Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
        Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals
    WARNING:[ 5 ]
        Clock_info18 [1] :Reports unconstrained ports summary
        Clock_sync05 [2] :Reports primary inputs sampled by multiple clock domains
        Clock_sync09 [1] :Reports signals synchronized more than once in the same clock domain
        Reset_sync01 [1] :Reports asynchronous reset signals which are not de-asserted synchronously
            Reset signal 'processor1.rst' for flop 'processor1.tmp_data[0]' is not synchronously de-asserted relative
            to clock signal 'processor1.clk1', ../../designA/design/processor1.v, 280
    INFO:[ 28 ]
```

[Tips]

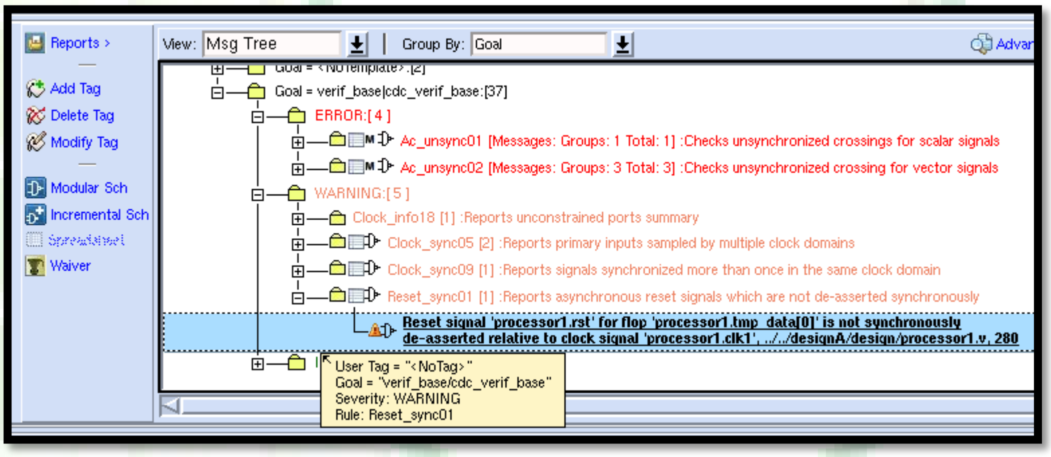
The rule “Reset_sync01” checks an asynchronous reset, should be synchronously de-asserted, relative to the clock. If an asynchronous reset is NOT synchronously de-asserted, the it may cause the following problems:

- Violation of the reset recovery time.
- Reset removal happening in different clock cycles for different sequential elements.

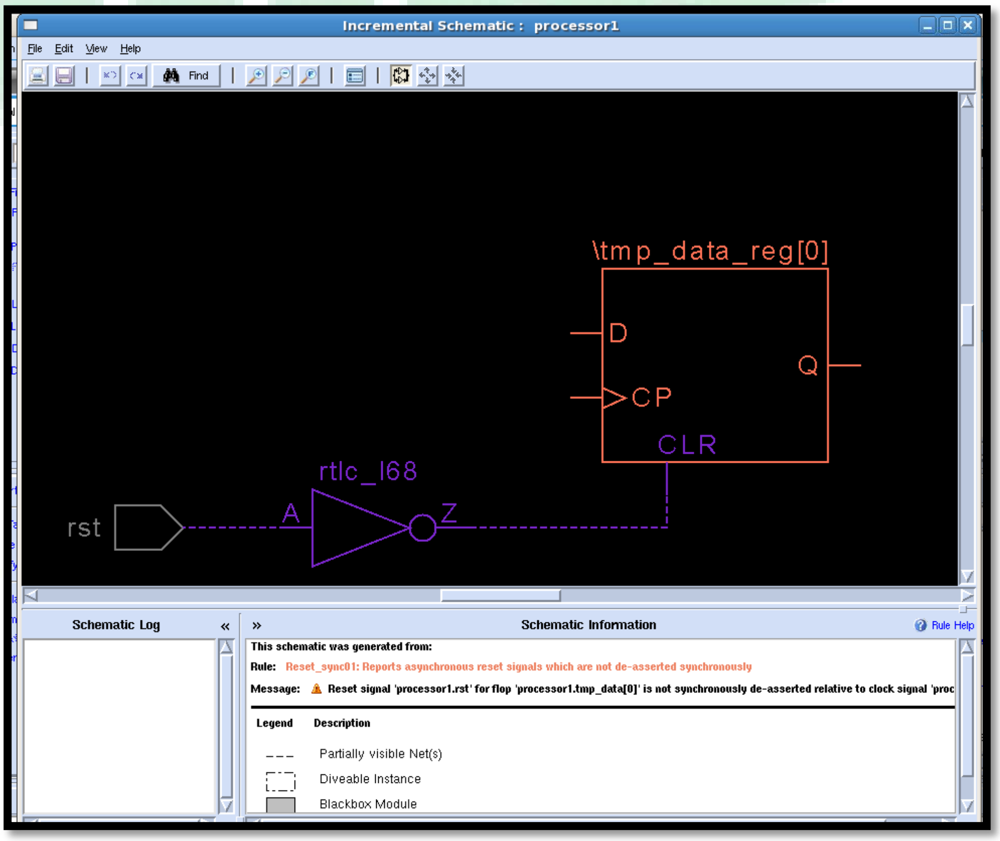
Synchronization helps reduce such possible problems in the design. We can achieve synchronization by various means. A simple approach is to pass the reset through a metastability structure.

20. Let us assume that the reset synchronization is coming from a reset synchronizer block which is NOT part of this design. If this is the case, then you will see the messages reported for all occurrences where a reset is NOT asynchronously asserted and the synchronously de-asserted. There messages could be considered as NOISE. One way to prevent SpyGlass from reporting these messages is to apply an input constraint on the reset port and associate it to a clock.

21. Double-clock the message for flip-flop “processor1.tmp_data[0]” in the “Reset_sync01” rule folder for “top.rst” and invoke the “Incremental Sch”.



```
Goal = <NoTemplate>:[2]
Goal = verif_base|cdc_verif_base:[37]
  ERROR:[ 4 ]
    Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
    Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals
  WARNING:[ 5 ]
    Clock_info18 [1] :Reports unconstrained ports summary
    Clock_sync05 [2] :Reports primary inputs sampled by multiple clock domains
    Clock_sync09 [1] :Reports signals synchronized more than once in the same clock domain
    Reset_sync01 [1] :Reports asynchronous reset signals which are not de-asserted synchronously
      Reset signal 'processor1.rst' for flop 'processor1.tmp_data[0]' is not synchronously
      de-asserted relative to clock signal 'processor1.clk1', ../../designA/design/processor1.v, 280

User Tag = "<NoTag>"
Goal = "verif_base/cdc_verif_base"
Severity: WARNING
Rule: Reset_sync01
```



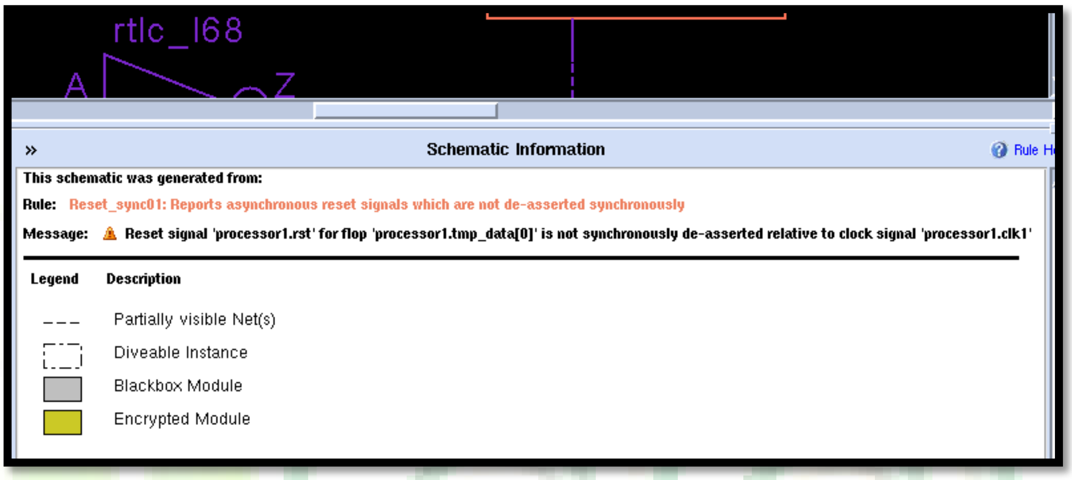
This schematic was generated from:

Rule: Reset_sync01: Reports asynchronous reset signals which are not de-asserted synchronously

Message: Reset signal 'processor1.rst' for flop 'processor1.tmp_data[0]' is not synchronously de-asserted relative to clock signal 'proc

Legend	Description
– – –	Partially visible Net(s)
⬚	Diveable Instance
▨	Blackbox Module

22. The message report that the reset signal is NOT being synchronously de-asserted relative to clock signal “clk1”.



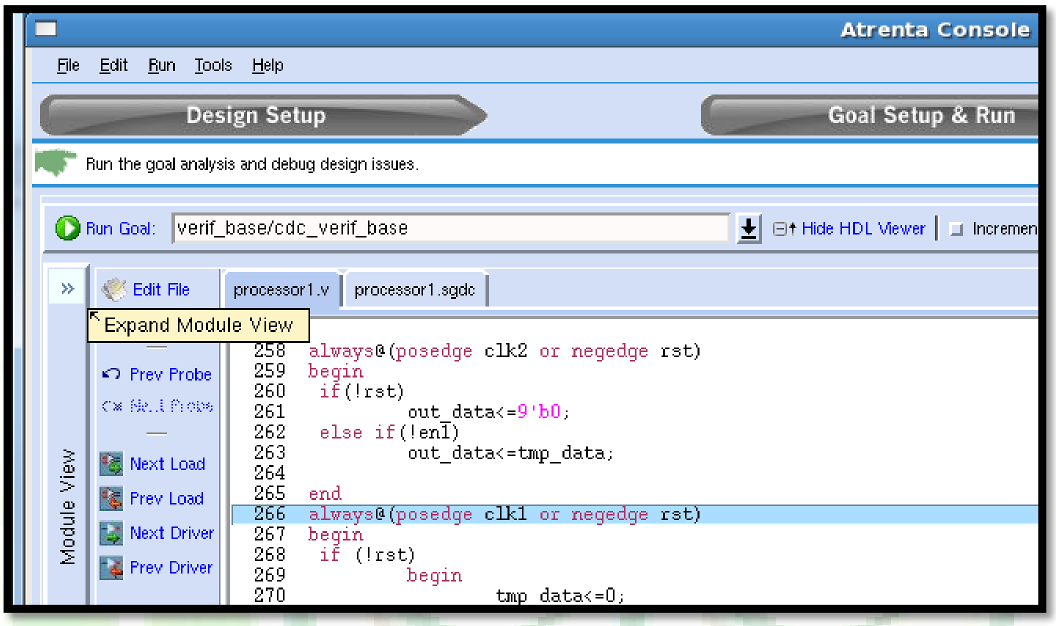
This schematic was generated from:

Rule: Reset_sync01: Reports asynchronous reset signals which are not de-asserted synchronously

Message: ⚠ Reset signal 'processor1.rst' for flop 'processor1.tmp_data[0]' is not synchronously de-asserted relative to clock signal 'processor1.clk1'

Legend	Description
---	Partially visible Net(s)
⬚	Diveable Instance
▨	Blackbox Module
🟨	Encrypted Module

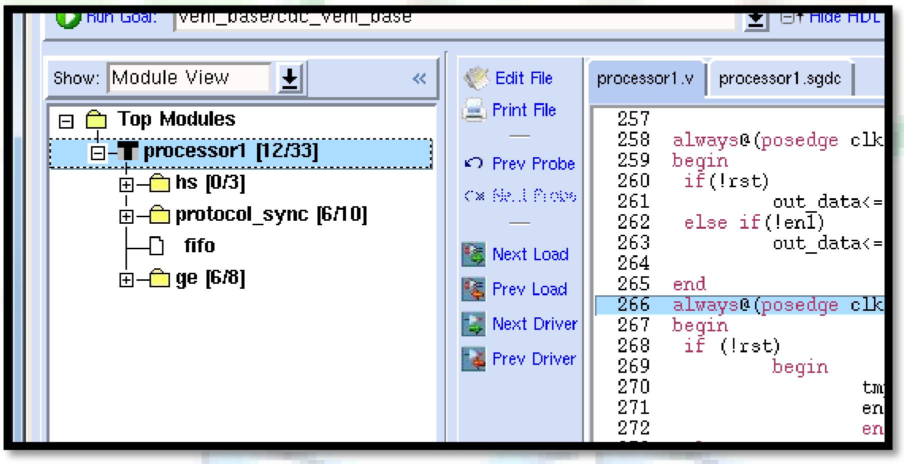
23. Let us apply an input constraint on the “rst” pin. At “Atrenta Console” window, expand the “Module View” by click “>>”



Run the goal analysis and debug design issues.

Run Goal: verif_base/cdc_verif_base

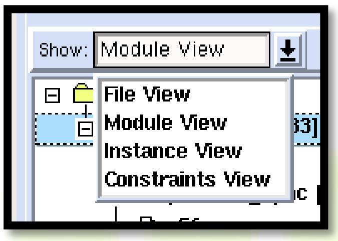
```
always@(posedge clk2 or negedge rst)
begin
 if(!rst)
        out_data<=9'b0;
 else if(!en1)
        out_data<=tmp_data;

end
always@(posedge clk1 or negedge rst)
begin
 if (!rst)
        begin
                tmp_data<=0;
```

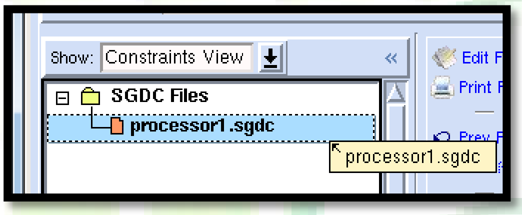


```
Top Modules
  processor1 [12/33]
    hs [0/3]
    protocol_sync [6/10]
    fifo
    ge [6/8]
```

```
always@(posedge clk
begin
 if(!rst)
        out_data<=
 else if(!en1)
        out_data<=

end
always@(posedge clk
begin
 if (!rst)
        begin
                tm
                en
                en
```

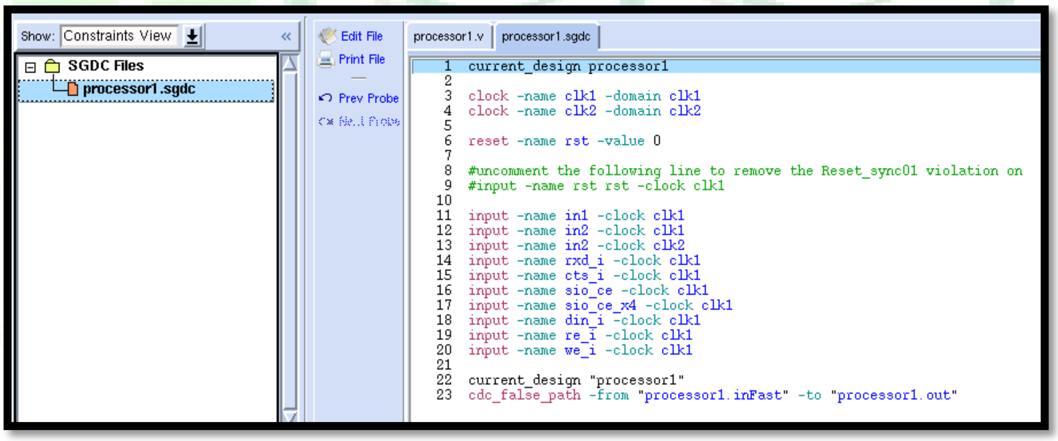
24. Switch Show type to “Constraints View”.





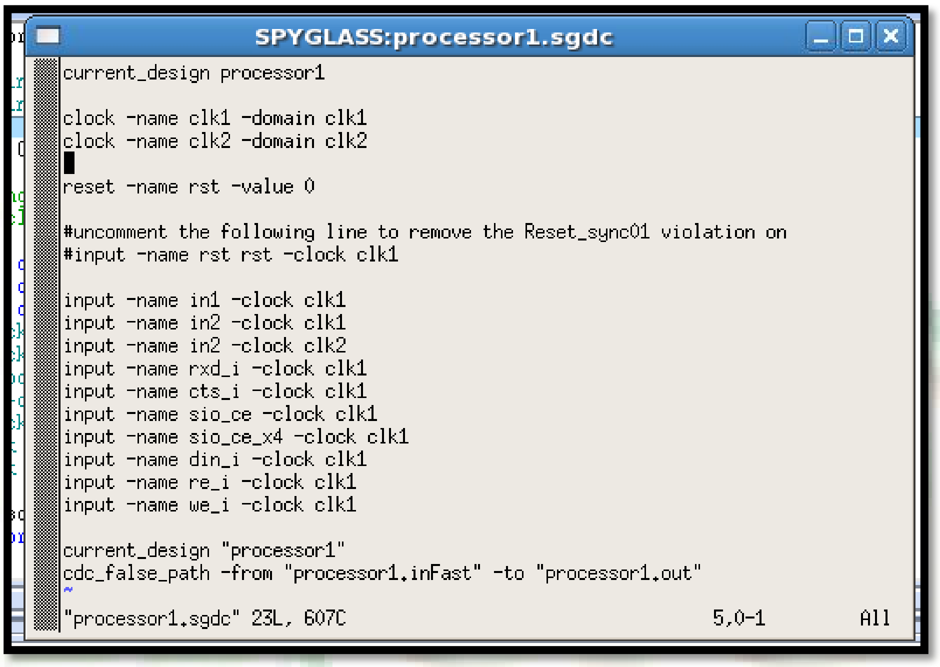
25. Double-clock the constrains (SGDC) file from console UI

SGDC file name	processor1.sgdc



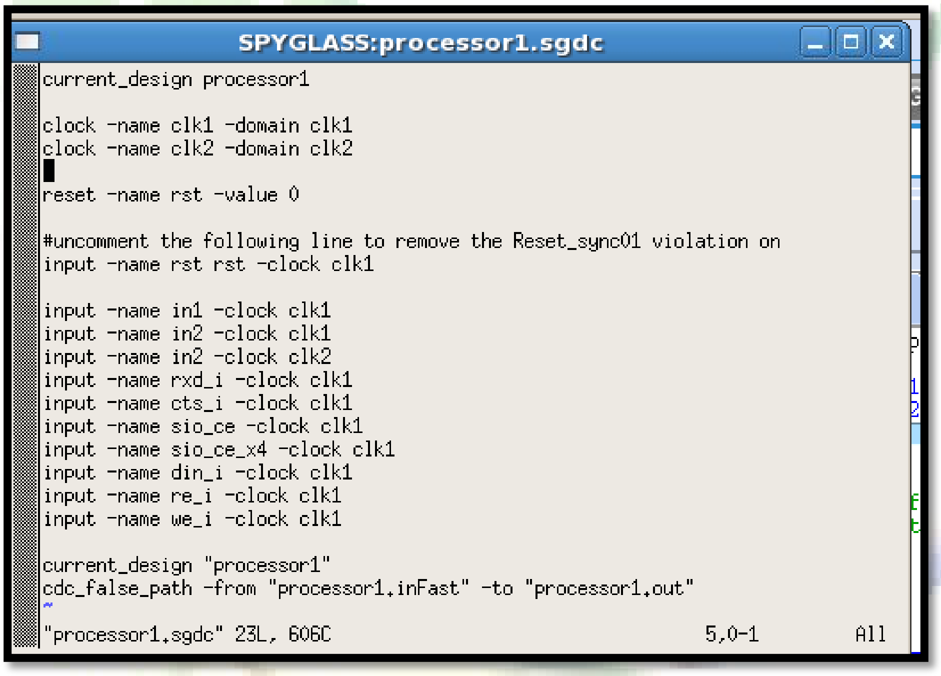
```
current_design processor1

clock -name clk1 -domain clk1
clock -name clk2 -domain clk2

reset -name rst -value 0

#uncomment the following line to remove the Reset_sync01 violation on
#input -name rst rst -clock clk1

input -name in1 -clock clk1
input -name in2 -clock clk1
input -name in2 -clock clk2
input -name rxd_i -clock clk1
input -name cts_i -clock clk1
input -name sio_ce -clock clk1
input -name sio_ce_x4 -clock clk1
input -name din_i -clock clk1
input -name re_i -clock clk1
input -name we_i -clock clk1

current_design "processor1"
cdc_false_path -from "processor1.inFast" -to "processor1.out"
```

26. Edit the “processor1.sgdc” constraint file by pressing the hotkey **<e>**. This will open the file in the editor window. Uncomment the line with input declared.



SPYGLASS:processor1.sgdc

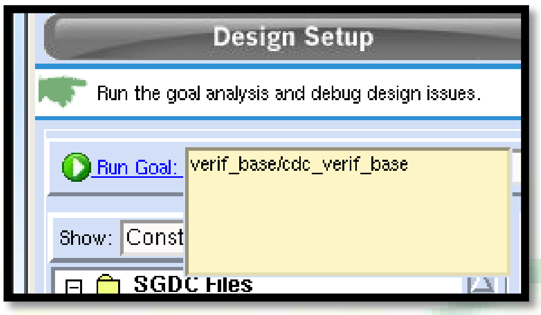
```
current_design processor1

clock -name clk1 -domain clk1
clock -name clk2 -domain clk2

reset -name rst -value 0

#uncomment the following line to remove the Reset_sync01 violation on
#input -name rst rst -clock clk1

input -name in1 -clock clk1
input -name in2 -clock clk1
input -name in2 -clock clk2
input -name rxd_i -clock clk1
input -name cts_i -clock clk1
input -name sio_ce -clock clk1
input -name sio_ce_x4 -clock clk1
input -name din_i -clock clk1
input -name re_i -clock clk1
input -name we_i -clock clk1

current_design "processor1"
cdc_false_path -from "processor1.inFast" -to "processor1.out"
~
"processor1.sgdc" 23L, 607C                              5,0-1          All
```



SPYGLASS:processor1.sgdc

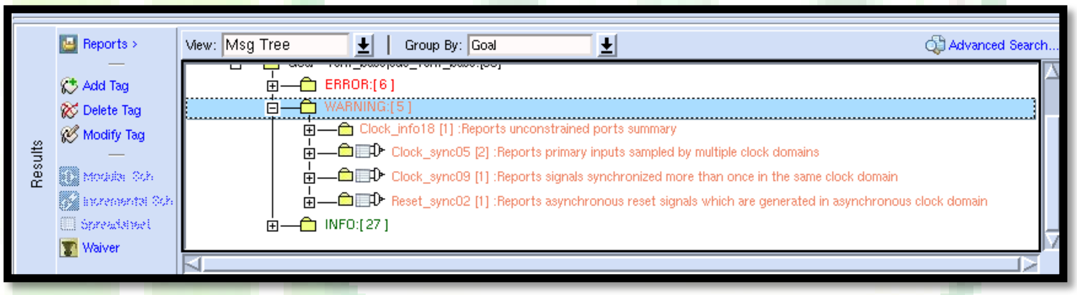
```
current_design processor1

clock -name clk1 -domain clk1
clock -name clk2 -domain clk2

reset -name rst -value 0

#uncomment the following line to remove the Reset_sync01 violation on
input -name rst rst -clock clk1

input -name in1 -clock clk1
input -name in2 -clock clk1
input -name in2 -clock clk2
input -name rxd_i -clock clk1
input -name cts_i -clock clk1
input -name sio_ce -clock clk1
input -name sio_ce_x4 -clock clk1
input -name din_i -clock clk1
input -name re_i -clock clk1
input -name we_i -clock clk1

current_design "processor1"
cdc_false_path -from "processor1.inFast" -to "processor1.out"
~
"processor1.sgdc" 23L, 606C                              5,0-1          All
```

27. Rerun the selected gaol



verif_base/cdc_verif_base

28. No further “Reset_sync01” messages are being reported for “rst” with respect to “clk1”.



```
ERROR:[6]
WARNING:[5]
    Clock_info18 [1] :Reports unconstrained ports summary
    Clock_sync05 [2] :Reports primary inputs sampled by multiple clock domains
    Clock_sync09 [1] :Reports signals synchronized more than once in the same clock domain
    Reset_sync02 [1] :Reports asynchronous reset signals which are generated in asynchronous clock domain
INFO:[27]
```

[Think about it]

If an input is registered in more than one clock domain, it will definitely be asynchronous with at least one of the clocks. Therefore, a synchronizer is required on at least on flop. The “Clock_sync05” rule is designed to flag these situations. Now, can you tell why primary input signal "in2" reported as "Clock_sync05" warning?

(HINT: Use the Msg Tree tab and Incremental schematic.)

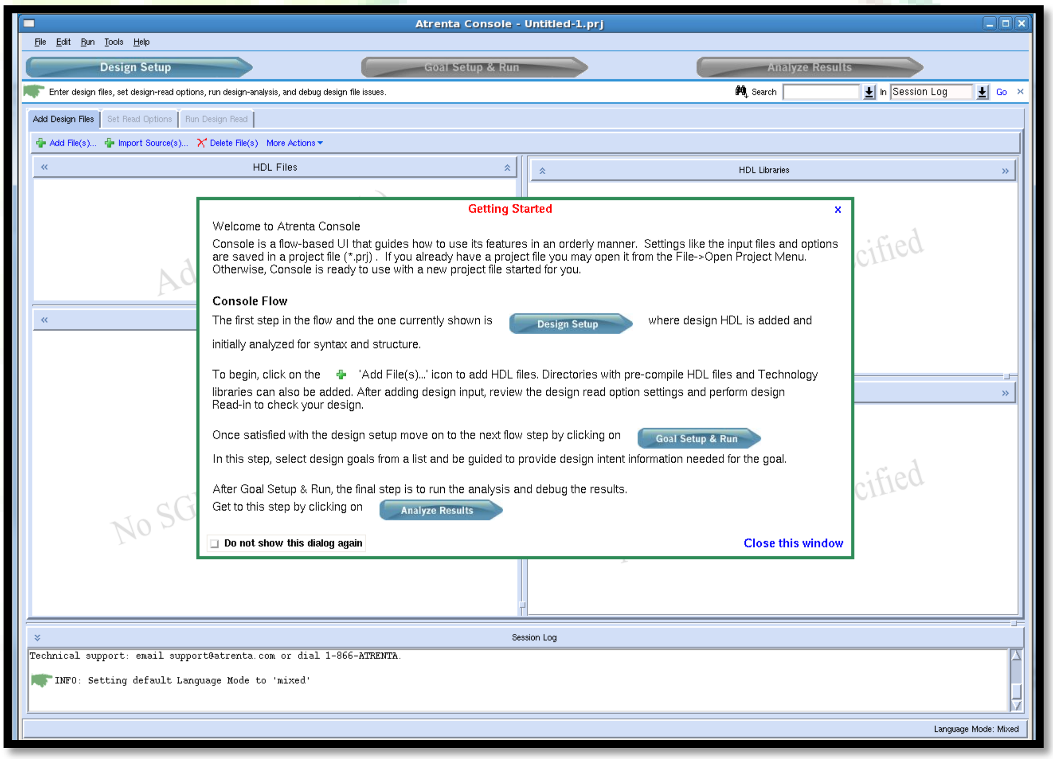
Lab 8 CDC functional verification

1. Change directory to lab 8

   Unix% cd ~/ADPLL_LAB_2012_Winter/LAB/Lab_8

2. Open the SpyGlass in interactive mode

   Unix% spyglass



Enter design files, set design-read options, run design-analysis, and debug design file issues.

Getting Started

Welcome to Atrenta Console

Console is a flow-based UI that guides how to use its features in an orderly manner. Settings like the input files and options are saved in a project file (*.prj) . If you already have a project file you may open it from the File->Open Project Menu. Otherwise, Console is ready to use with a new project file started for you.

Console Flow

The first step in the flow and the one currently shown is [Design Setup] where design HDL is added and initially analyzed for syntax and structure.

To begin, click on the 'Add File(s)...' icon to add HDL files. Directories with pre-compile HDL files and Technology libraries can also be added. After adding design input, review the design read option settings and perform design Read-in to check your design.

Once satisfied with the design setup move on to the next flow step by clicking on [Goal Setup & Run]

In this step, select design goals from a list and be guided to provide design intent information needed for the goal.

After Goal Setup & Run, the final step is to run the analysis and debug the results.
Get to this step by clicking on [Analyze Results]

Do not show this dialog again

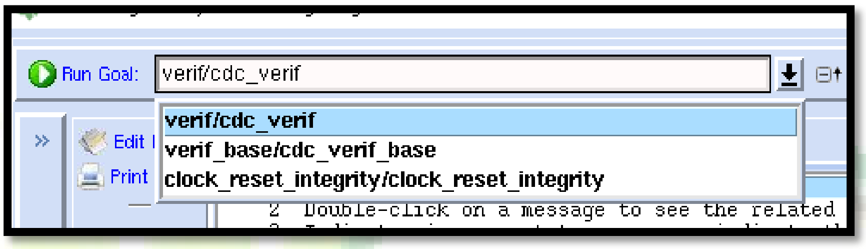
Close this window

```
Technical support: email support@atrenta.com or dial 1-866-ATRENTA.
INFO: Setting default Language Mode to 'mixed'
```

3. Open project

Project file name	Lab8.prj

4. Once you've cleaned up all the metastability issues, you are now ready to perform advance CDC checks on the design, such as Reset verification, FIFO/Handshake verification, etc. Click the drop down selection to load results of "cdc_verif" goal.



Run Goal: verif/cdc_verif

```
verif/cdc_verif
verif_base/cdc_verif_base
clock_reset_integrity/clock_reset_integrity
```

[Think about it]

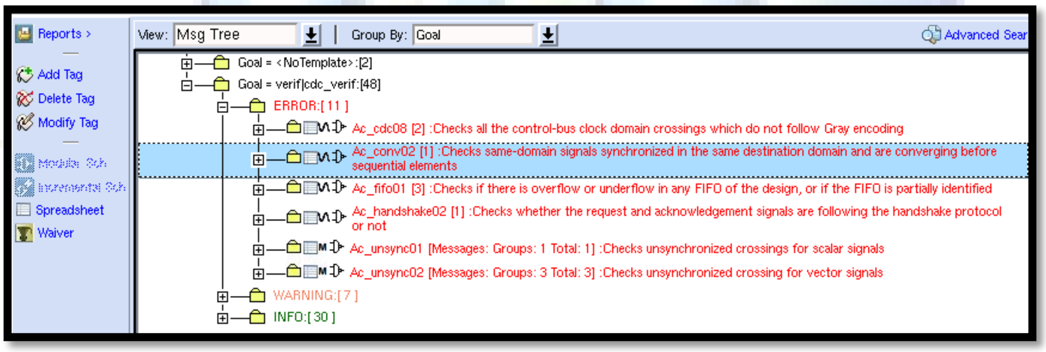
In addition to performing structural checks SpyGlass CDC will perform some functional checks using formal engines to verify the design functionality. Even if your synchronization schemes are in place, how sure can you be that they are functioning as expected? For example you have implemented a FIFO synchronizing scheme, how can you be sure that FIFO does not under-run or overflow? You have implemented some control logic which makes use of a gray code counter. How sure can you be that the behavior is as expected?

[Tips]

Make sure you do NOT have any "Ac_sanity04" messages. If there is any "Ac_sanity04", none of the rest of the "Ac_" rules would be run due to this "Ac_sanity04"

5. Data coherency (Convergence) – Gray code checks

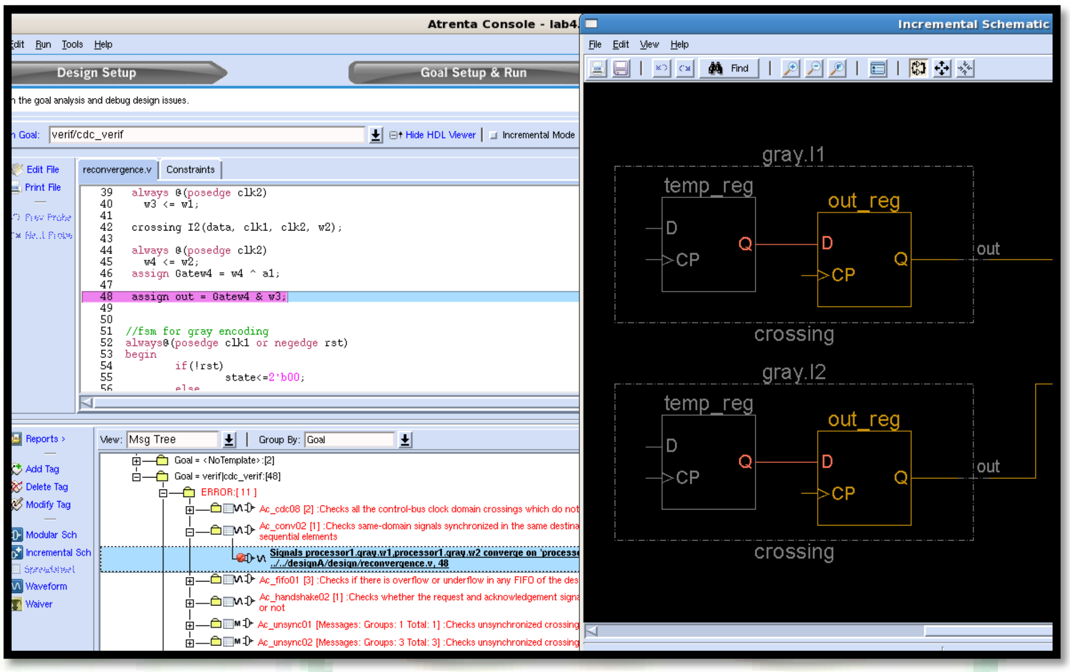
In the "Msg tree" tab of the message window, expand the "verif/cdc_verif" goal folder and then expand the "Ac_conv02" rule folder.



View: Msg Tree   Group By: Goal

```
Goal = <NoTemplate>:[2]
Goal = verif|cdc_verif:[48]
  ERROR:[ 11 ]
    Ac_cdc08 [2] :Checks all the control-bus clock domain crossings which do not follow Gray encoding
    Ac_conv02 [1] :Checks same-domain signals synchronized in the same destination domain and are converging before sequential elements
    Ac_fifo01 [3] :Checks if there is overflow or underflow in any FIFO of the design, or if the FIFO is partially identified
    Ac_handshake02 [1] :Checks whether the request and acknowledgement signals are following the handshake protocol or not
    Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
    Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals
  WARNING:[ 7 ]
  INFO:[ 30 ]
```

Double-click the reported violation message and use hotkey **<i>** to open incremental schematic window.

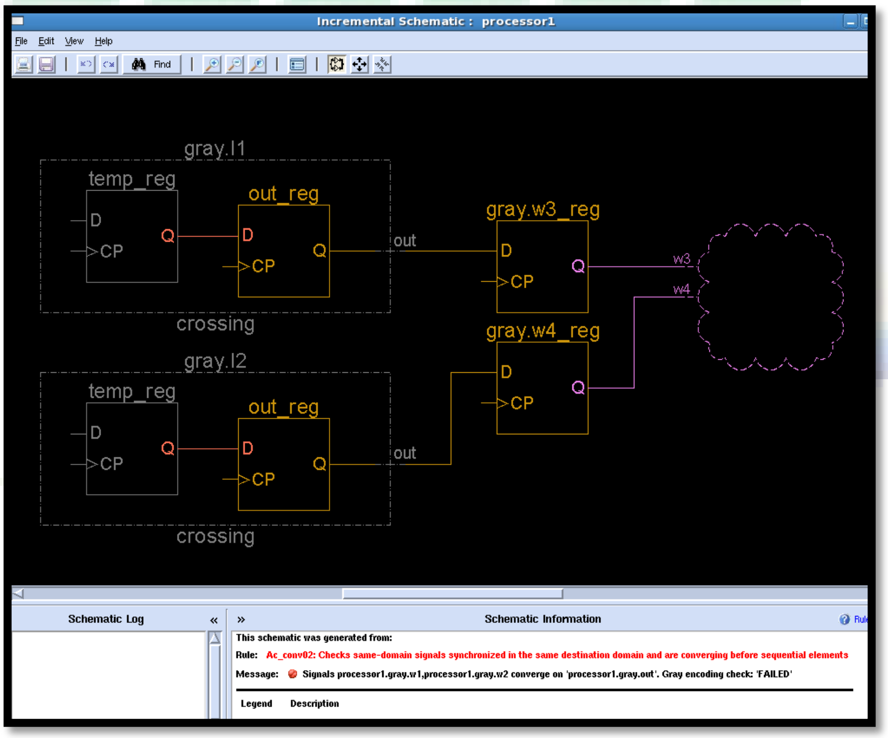


n the goal analysis and debug design issues.

```
  always @(posedge clk2)
    w3 <= w1;

  crossing I2(data, clk1, clk2, w2);

  always @(posedge clk2)
    w4 <= w2;
  assign Gatew4 = w4 ^ a1;

  assign out = Gatew4 & w3;


 //fsm for gray encoding
 always@(posedge clk1 or negedge rst)
 begin
         if(!rst)
                 state<=2'b00;
         else
```

```
Goal = <NoTemplate>:[2]
Goal = verif|cdc_verif:[48]
  ERROR:[ 11 ]
    Ac_cdc08 [2] :Checks all the control-bus clock domain crossings which do not
    Ac_conv02 [1] :Checks same-domain signals synchronized in the same destina
    sequential elements
      Signals processor1.gray.w1,processor1.gray.w2 converge on 'processo
      ../../designA/design/reconvergence.v, 48
    Ac_fifo01 [3] :Checks if there is overflow or underflow in any FIFO of the des
    Ac_handshake02 [1] :Checks whether the request and acknowledgement signa
    or not
    Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossing
    Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing
```

When the synchronized signals converge, they typically, but NOT always, originate from some type of gray code counter. Once determining the re-converging synchronized signals, advance CDC checks can perform a gray code check on the signals.



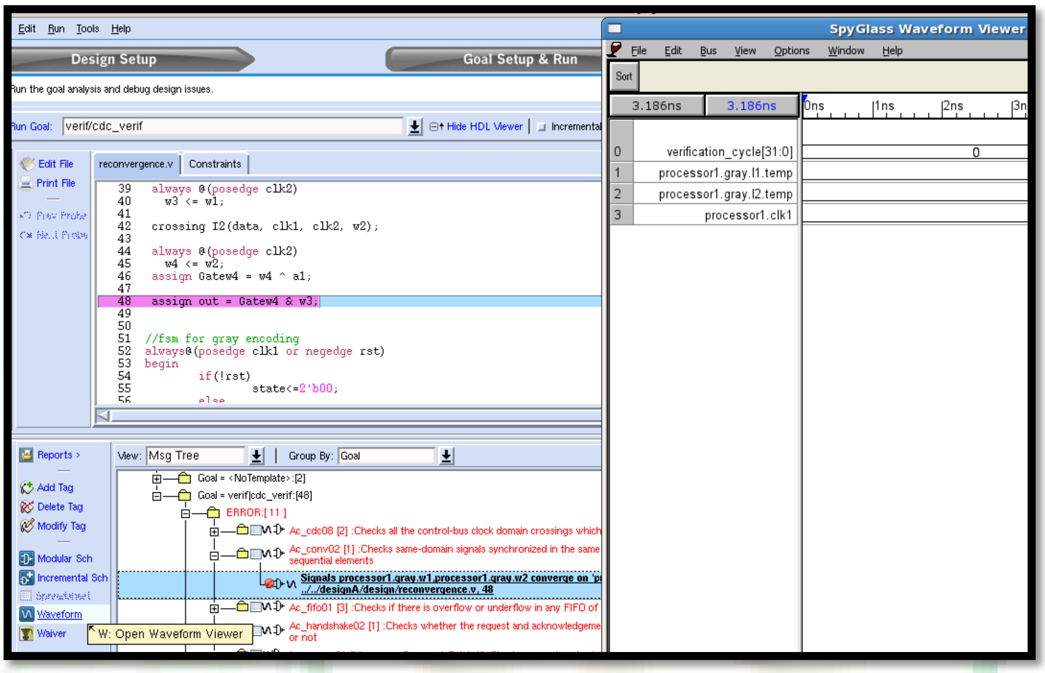
This schematic was generated from:

Rule: Ac_conv02: Checks same-domain signals synchronized in the same destination domain and are converging before sequential elements

Message: Signals processor1.gray.w1,processor1.gray.w2 converge on 'processor1.gray.out'. Gray encoding check: 'FAILED'

Legend Description

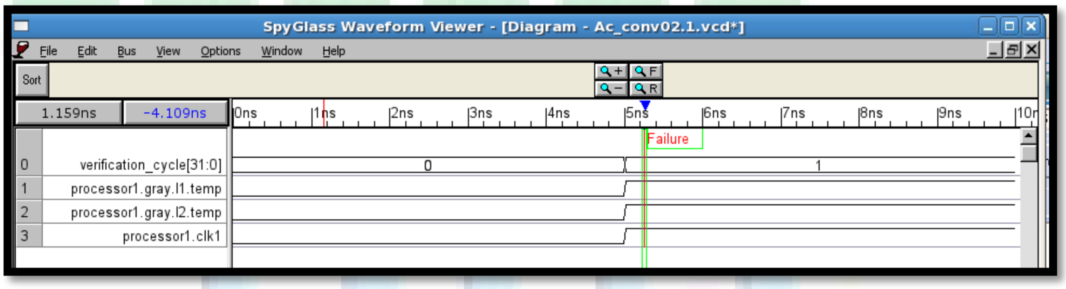
With the message highlighted within "Ac_conv02" rule folder, use hotkey <w> to open the "SpyGlass Waveform Viewer".



```
always @(posedge clk2)
  w3 <= w1;

crossing I2(data, clk1, clk2, w2);

always @(posedge clk2)
  w4 <= w2;
assign Gatew4 = w4 ^ a1;

assign out = Gatew4 & w3;


//fsm for gray encoding
always@(posedge clk1 or negedge rst)
begin
        if(!rst)
                state<=2'b00;
        else
```

```
Goal = <NoTemplate>:[2]
Goal = verif|cdc_verif:[48]
  ERROR:[11]
    Ac_cdc08 [2] :Checks all the control-bus clock domain crossings which
    Ac_conv02 [1] :Checks same-domain signals synchronized in the same sequential elements
      Signals processor1.gray.w1,processor1.gray.w2 converge on 'p
      ../../designA/design/reconvergence.v, 48
    Ac_fifo01 [3] :Checks if there is overflow or underflow in any FIFO of
    Ac_handshake02 [1] :Checks whether the request and acknowledgeme or not
```

From the waveform, it can be seen that signals feeding destination of 2 crossings "process1.gray.I1.temp" and "processor1.gray.I2.temp" are changing at the same time. Hence, this is a violation of gray code.


SpyGlass Waveform Viewer - [Diagram - Ac_conv02.1.vcd*]
File Edit Bus View Options Window Help
Sort
1.159ns
-4.109ns
0ns
1ns
2ns
3ns
4ns
5ns
6ns
7ns
8ns
9ns
10n
Failure
0 verification_cycle[31:0]
1 processor1.gray.I1.temp
2 processor1.gray.I2.temp
3 processor1.clk1
0
1

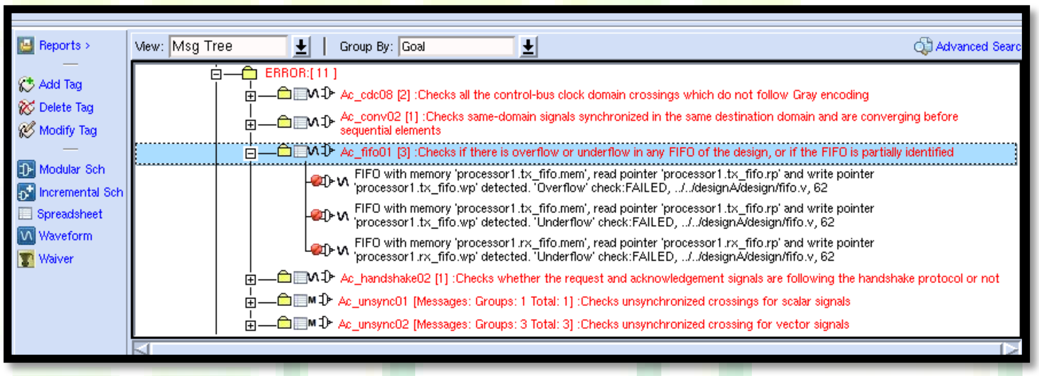
[Think about it]

Why the SpyGlass report re-converging singals do not follow gray encoding?

6. Close all windows expect the “Atreanta Console” window.

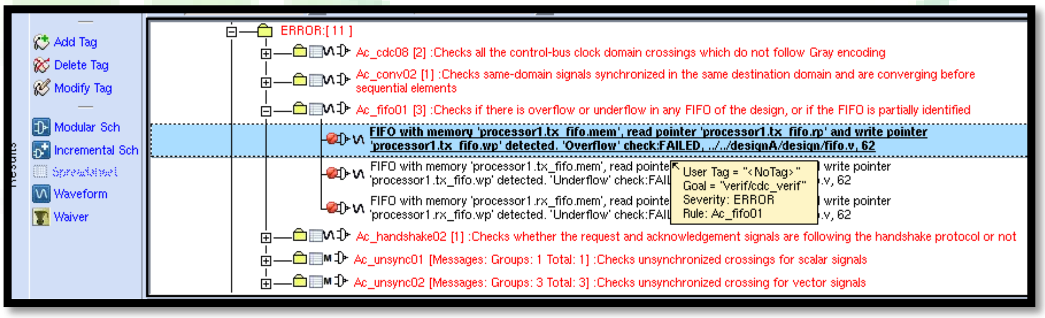
7. Complex synchronizations schemes: FIFO under-run and overflow checks.

   In the “Msg tree” tab of the message window, expend the “verif/cdc_verif” goal folder and then expand the “Ac_fifo01” rule folder.



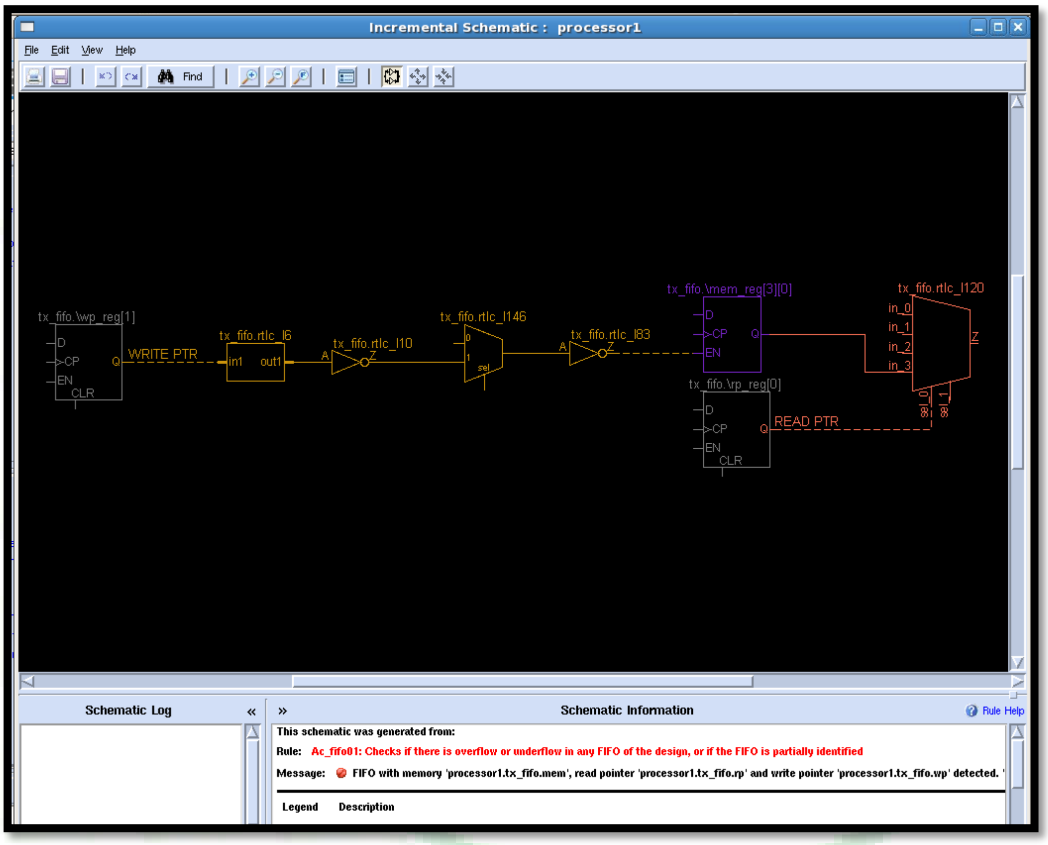
```
ERROR:[ 11 ]
  Ac_cdc08 [2] :Checks all the control-bus clock domain crossings which do not follow Gray encoding
  Ac_conv02 [1] :Checks same-domain signals synchronized in the same destination domain and are converging before
  sequential elements
  Ac_fifo01 [3] :Checks if there is overflow or underflow in any FIFO of the design, or if the FIFO is partially identified
    FIFO with memory 'processor1.tx_fifo.mem', read pointer 'processor1.tx_fifo.rp' and write pointer
    'processor1.tx_fifo.wp' detected. 'Overflow' check:FAILED, ../../designA/design/fifo.v, 62
    FIFO with memory 'processor1.tx_fifo.mem', read pointer 'processor1.tx_fifo.rp' and write pointer
    'processor1.tx_fifo.wp' detected. 'Underflow' check:FAILED, ../../designA/design/fifo.v, 62
    FIFO with memory 'processor1.rx_fifo.mem', read pointer 'processor1.rx_fifo.rp' and write pointer
    'processor1.rx_fifo.wp' detected. 'Underflow' check:FAILED, ../../designA/design/fifo.v, 62
  Ac_handshake02 [1] :Checks whether the request and acknowledgement signals are following the handshake protocol or not
  Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
  Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals
```

Double-clock the message with memory “processor1.tx_fifo.mem” specifically the one with overflow. The message informs us that the read and write pointers have been identified for this FIFO implementation.



```
User Tag = "<NoTag>"
Goal = "verif/cdc_verif"
Severity: ERROR
Rule: Ac_fifo01
```

Invoke incremental schematic by pressing hotkey **<i>** to see identified FIFO structure.



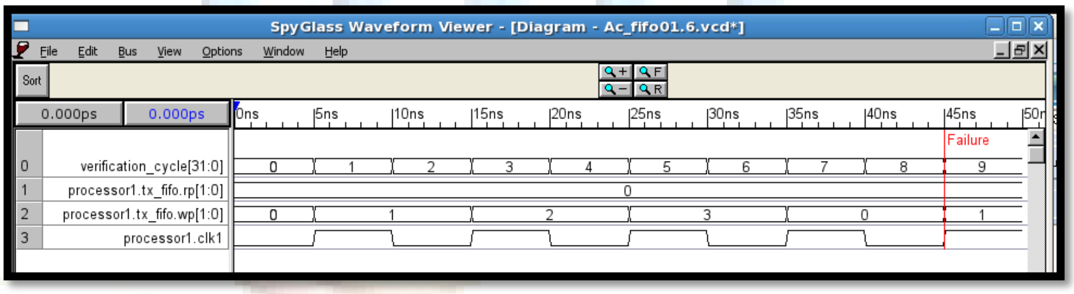
This schematic was generated from:

Rule: Ac_fifo01: Checks if there is overflow or underflow in any FIFO of the design, or if the FIFO is partially identified

Message: FIFO with memory 'processor1.tx_fifo.mem', read pointer 'processor1.tx_fifo.rp' and write pointer 'processor1.tx_fifo.wp' detected. '

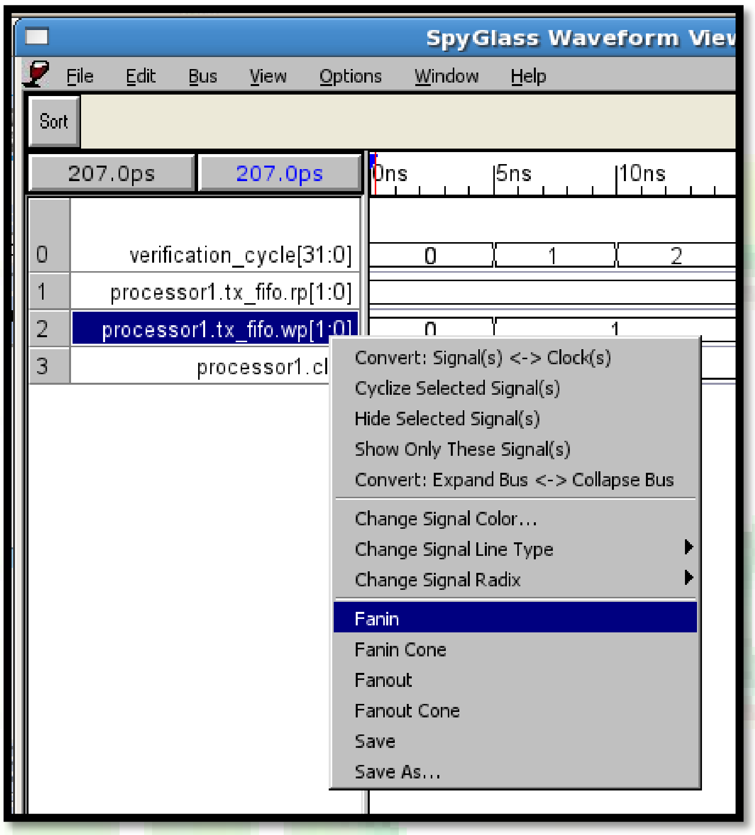
Legend Description

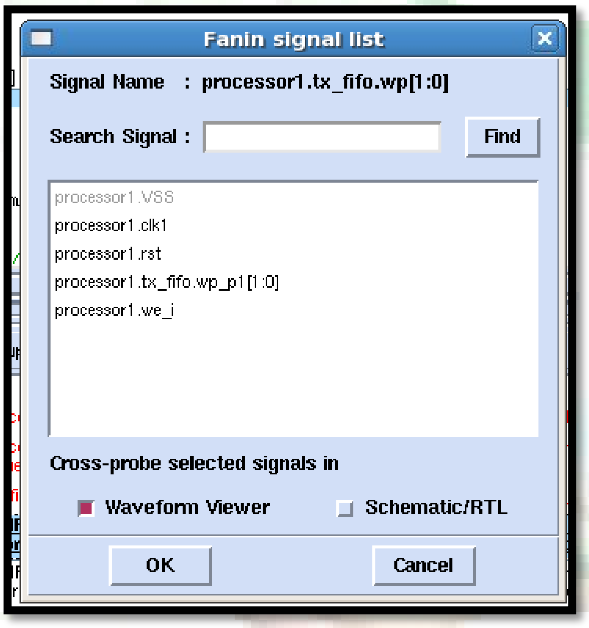
Advanced CDC checks can go one step further and identify whether the FIFO under-runs or overflows. The selected message reports that an overflow conditions has occurred. To view the failure, open the waveform, it can be inferred that the write pointer has been re-written into the address location 0. Now, at “Msg Tree” tab press hotkey **<w>** to open “SpyGlass Waveform Viewer”.



SpyGlass Waveform Viewer - [Diagram - Ac_fifo01.6.vcd*]
File Edit Bus View Options Window Help
Sort
0.000ps
0.000ps
0ns 5ns 10ns 15ns 20ns 25ns 30ns 35ns 40ns 45ns 50n
Failure
0 verification_cycle[31:0] 0 1 2 3 4 5 6 7 8 9
1 processor1.tx_fifo.rp[1:0] 0
2 processor1.tx_fifo.wp[1:0] 0 1 2 3 0 1
3 processor1.clk1

Right-click on “processor1.tx_fifo.wp[1:0]” and select “fanin”.


SpyGlass Waveform View
File Edit Bus View Options Window Help
Sort
207.0ps
207.0ps
0ns
5ns
10ns
0 verification_cycle[31:0]
1 processor1.tx_fifo.rp[1:0]
2 processor1.tx_fifo.wp[1:0]
3 processor1.cl
0
1
2
0
1
Convert: Signal(s) <-> Clock(s)
Cyclize Selected Signal(s)
Hide Selected Signal(s)
Show Only These Signal(s)
Convert: Expand Bus <-> Collapse Bus
Change Signal Color…
Change Signal Line Type
Change Signal Radix
Fanin
Fanin Cone
Fanout
Fanout Cone
Save
Save As…



Signal Name : processor1.tx_fifo.wp[1:0]

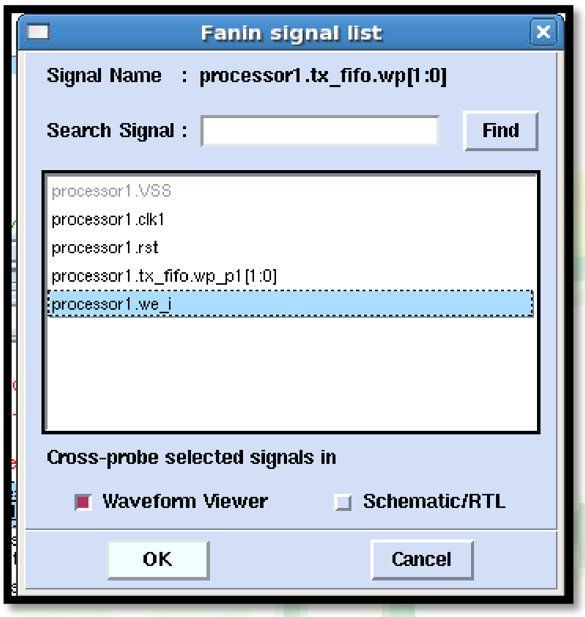
Search Signal :

processor1.VSS
processor1.clk1
processor1.rst
processor1.tx_fifo.wp_p1[1:0]
processor1.we_i

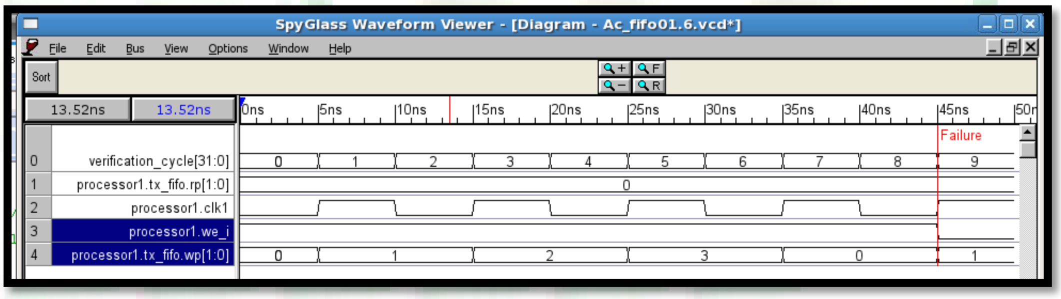
Cross-probe selected signals in

[✓ Waveform Viewer] Schematic/RTL

Click to select "we_i" signals and press "OK".



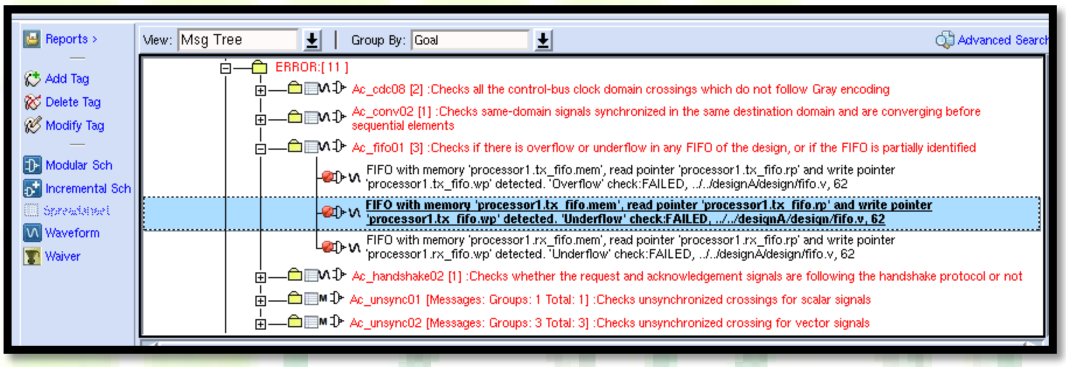
It is apparent that write enable (we_i) is active even the FIFO is full.


SpyGlass Waveform Viewer - [Diagram - Ac_fifo01.6.vcd*]
File Edit Bus View Options Window Help
Sort
13.52ns 13.52ns
0ns 5ns 10ns 15ns 20ns 25ns 30ns 35ns 40ns 45ns 50n
Failure
0 verification_cycle[31:0]
1 processor1.tx_fifo.rp[1:0]
2 processor1.clk1
3 processor1.we_i
4 processor1.tx_fifo.wp[1:0]

8. Close all windows expect the "Atreanta Console" window.

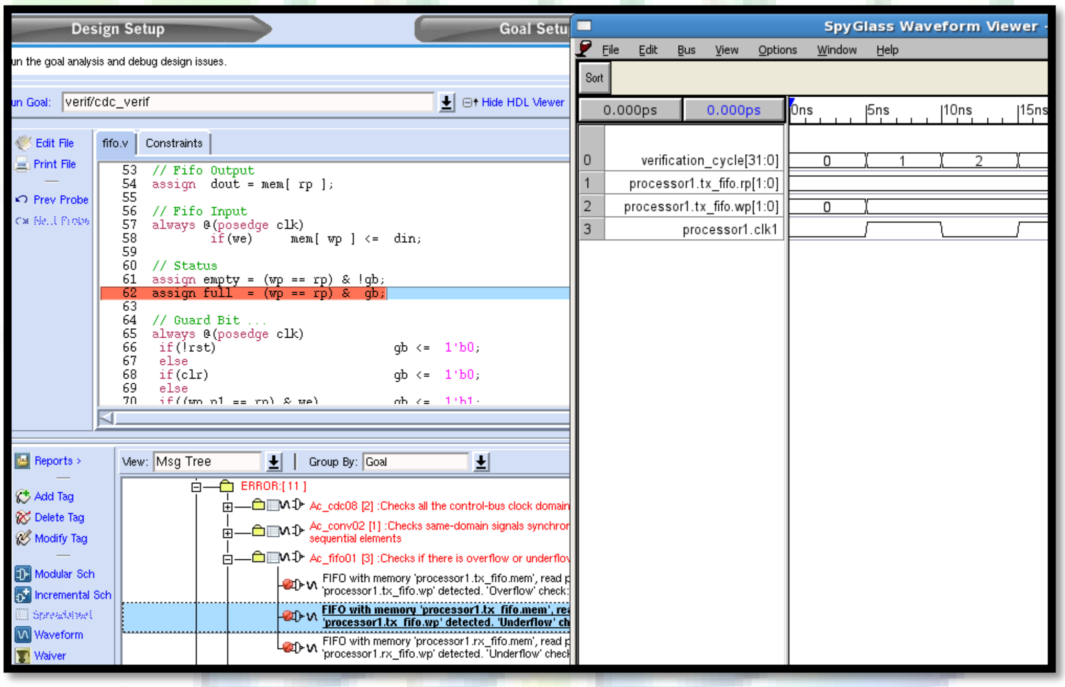
9. Complex synchronizations schemes: FIFO underflow checks

Double-click the message reported by the "Ac_fifo01" rule about the FIFO with memory "processor1.tx_fifo.mem". The message reports that an underflow condition has occurred.



```
View: Msg Tree    Group By: Goal    Advanced Search
ERROR:[11]
  Ac_cdc08 [2] :Checks all the control-bus clock domain crossings which do not follow Gray encoding
  Ac_conv02 [1] :Checks same-domain signals synchronized in the same destination domain and are converging before sequential elements
  Ac_fifo01 [3] :Checks if there is overflow or underflow in any FIFO of the design, or if the FIFO is partially identified
    FIFO with memory 'processor1.tx_fifo.mem', read pointer 'processor1.tx_fifo.rp' and write pointer 'processor1.tx_fifo.wp' detected. 'Overflow' check:FAILED, ../../designA/design/fifo.v, 62
    FIFO with memory 'processor1.tx_fifo.mem', read pointer 'processor1.tx_fifo.rp' and write pointer 'processor1.tx_fifo.wp' detected. 'Underflow' check:FAILED, ../../designA/design/fifo.v, 62
    FIFO with memory 'processor1.rx_fifo.mem', read pointer 'processor1.rx_fifo.rp' and write pointer 'processor1.rx_fifo.wp' detected. 'Underflow' check:FAILED, ../../designA/design/fifo.v, 62
  Ac_handshake02 [1] :Checks whether the request and acknowledgement signals are following the handshake protocol or not
  Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
  Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals
```

Open the waveform for this message.



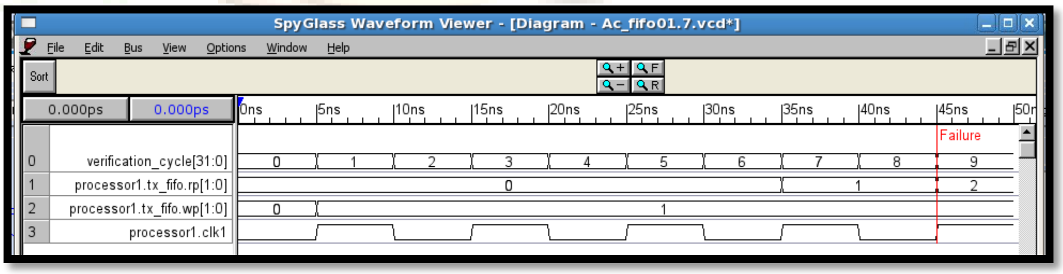
un the goal analysis and debug design issues.

un Goal: verif/cdc_verif

```
53  // Fifo Output
54  assign  dout = mem[ rp ];
55
56  // Fifo Input
57  always @(posedge clk)
58          if(we)     mem[ wp ] <=  din;
59
60  // Status
61  assign empty = (wp == rp) & !gb;
62  assign full  = (wp == rp) &  gb;
63
64  // Guard Bit ...
65  always @(posedge clk)
66   if(!rst)                        gb <=  1'b0;
67   else
68   if(clr)                         gb <=  1'b0;
69   else
```

```
0.000ps   0.000ps   0ns   5ns   10ns   15ns
0  verification_cycle[31:0]   0 | 1 | 2
1  processor1.tx_fifo.rp[1:0]
2  processor1.tx_fifo.wp[1:0]  0
3  processor1.clk1
```

Trace the "processor1.tx_fifo.rp[1:0]" read pointer signal.



SpyGlass Waveform Viewer - [Diagram - Ac_fifo01.7.vcd*]

```
0.000ps   0.000ps   0ns  5ns  10ns  15ns  20ns  25ns  30ns  35ns  40ns  45ns  50n
                                                                   Failure
0  verification_cycle[31:0]   0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
1  processor1.tx_fifo.rp[1:0] 0                                | 1 | 2
2  processor1.tx_fifo.wp[1:0] 0 | 1
3  processor1.clk1
```

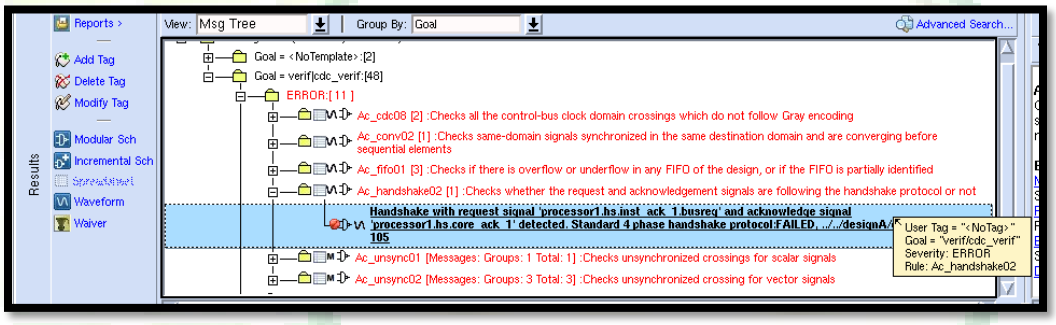
[Think about it]

What wrong with the “processor1.tx_fifo.rp[1:0]” read pointer signal?

10. Close all windows expect the “Atreanta Console” window.

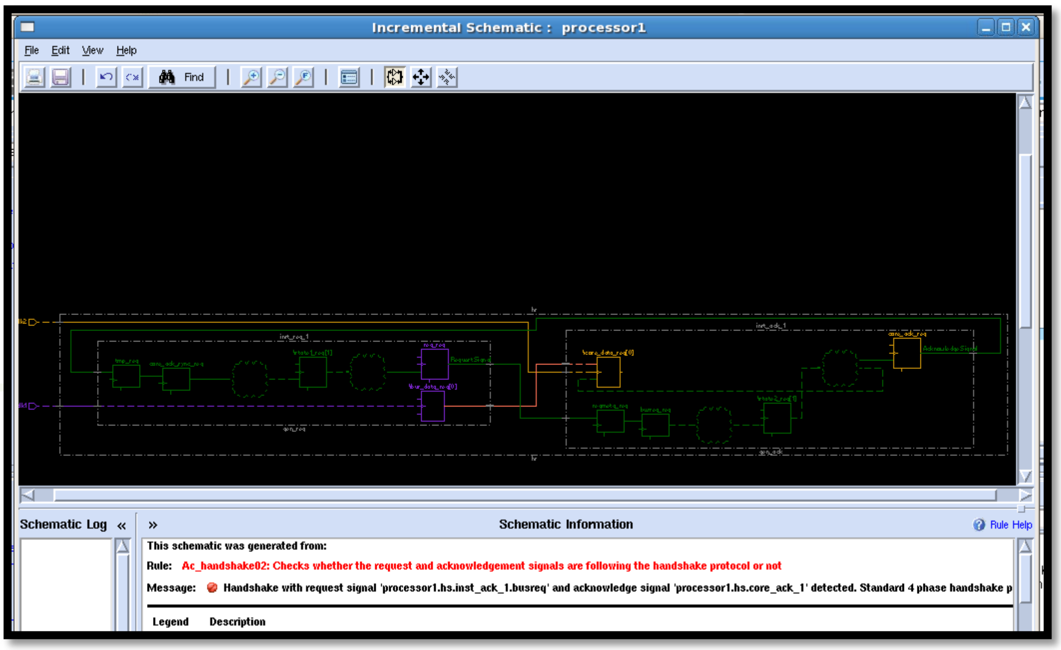
11. Handshake synchronizations schemes: data loss in handshake and four phase handshake protocol check.

In the message tree of the tab of the message window, expand the “verif” goal folder then expand the “Ac_handshake02” rule folder.



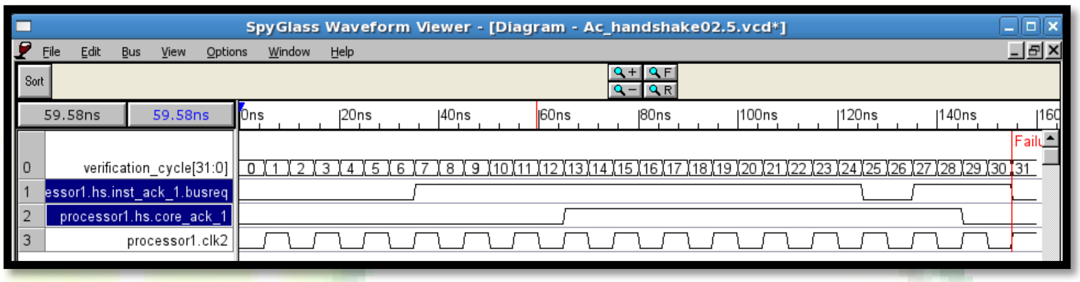
```
Goal = <NoTemplate>:[2]
Goal = verif|cdc_verif:[48]
  ERROR:[ 11 ]
    Ac_cdc08 [2] :Checks all the control-bus clock domain crossings which do not follow Gray encoding
    Ac_conv02 [1] :Checks same-domain signals synchronized in the same destination domain and are converging before sequential elements
    Ac_fifo01 [3] :Checks if there is overflow or underflow in any FIFO of the design, or if the FIFO is partially identified
    Ac_handshake02 [1] :Checks whether the request and acknowledgement signals are following the handshake protocol or not
      Handshake with request signal 'processor1.hs.inst_ack_1.busreq' and acknowledge signal 'processor1.hs.core_ack_1' detected. Standard 4 phase handshake protocol:FAILED, ../../designA/ 105
    Ac_unsync01 [Messages: Groups: 1 Total: 1] :Checks unsynchronized crossings for scalar signals
    Ac_unsync02 [Messages: Groups: 3 Total: 3] :Checks unsynchronized crossing for vector signals

User Tag = "<NoTag>"
Goal = "verif/cdc_verif"
Severity: ERROR
Rule: Ac_handshake02
```

Double click the message. This message informs that the request and acknowledgement signals have been identified for handshake synchronization structure. Advanced CDC checks whether the handshake synchronization follows four phase handshake protocol. Spyglass identifies the “REQ-ACK” loop of handshake structure. To view the “REQ-ACK” loop. Highlight the message by double clicking the message and then open incremental sch.



```
This schematic was generated from:
Rule:    Ac_handshake02: Checks whether the request and acknowledgement signals are following the handshake protocol or not
Message:    Handshake with request signal 'processor1.hs.inst_ack_1.busreq' and acknowledge signal 'processor1.hs.core_ack_1' detected. Standard 4 phase handshake p
Legend    Description
```

To view the failures open the waveform viewer. From the waveform viewer it can be seen that four phase protocol is not followed. The “busreq” signal goes inactive and then comes active again before the “core_ack_1” (acknowledgement) goes inactive.


SpyGlass Waveform Viewer - [Diagram - Ac_handshake02.5.vcd*]
File
Edit
Bus
View
Options
Window
Help
Sort
59.58ns
59.58ns
0ns
20ns
40ns
60ns
80ns
100ns
120ns
140ns
160
Failu
0 verification_cycle[31:0]
1 essor1.hs.inst_ack_1.busreq
2 processor1.hs.core_ack_1
3 processor1.clk2