SpyGlass® CDC

Customer Training

Release 5.3.0

This presentation may contain forward-looking statements regarding product development. Information or statements contained in this presentation are for informational purposes only and do not represent a commitment, promise, or legal obligation of any kind by Atrenta Inc. or its affiliates.

SpyGlass User Training Tracks

Getting Started with SpyGlass	Pre-Requisites: None - Recommended for all users
Lint & SoC Lint	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass Advanced Lint	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass CDC	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass Power Verify	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass Power Estimate/Reduce	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass Constraints	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass TXV	Pre-Requisites: SpyGlass Constraints
SpyGlass Physical Base	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass DFT & DSM	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass MBIST	Pre-Requisites: Getting Started with SpyGlass or equivalent
SpyGlass Tcl Interface	Pre-Requisites: Getting Started with SpyGlass or equivalent

Atrenta Confidential © 2014 Atrenta Inc.

Agenda

An Introduction to Clock Domain Crossing

The SpyGlass CDC solution

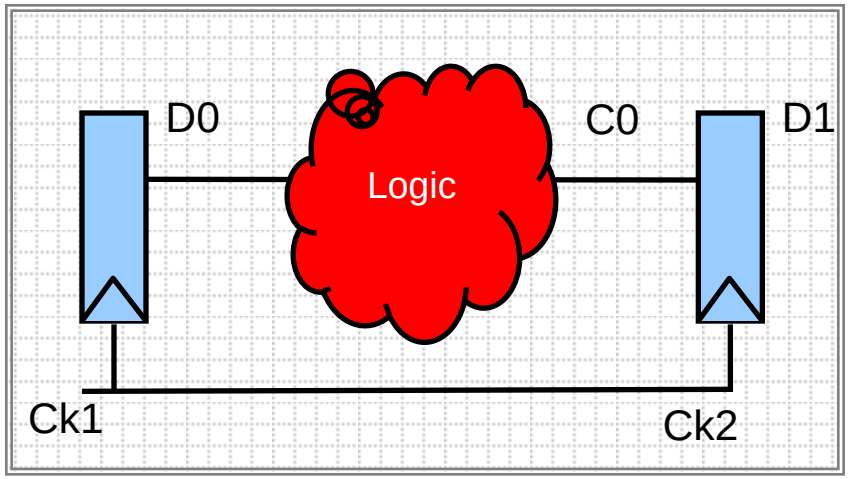
- CDC setup
- CDC setup check
- CDC structural verification
- CDC functional verification

Appendix

- Atrenta support information

Atrenta Confidential © 2014 Atrenta Inc.

Principles of Synchronous Design



Input to combination logic changes upon SOURCE clock edge

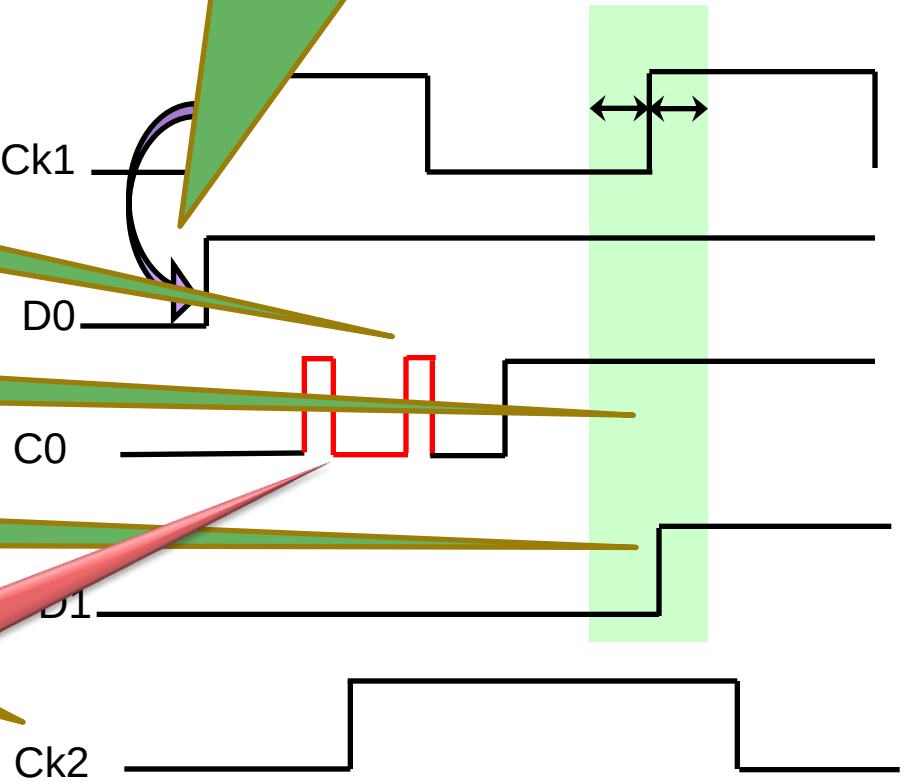
Combo logic creates GLITCHES

GLITCHES SETTLE before edge of SYNCHRONOUS clock

D1 captures stable data assuming setup/hold time is met. Clock domains are SYNCHRONOUS

But what if D1 is clocked by an ASYNCHRONOUS clock where setup/hold time can't be guara

Asynchronous designs are NOT safe and will always eventually fail.
MTBF = 1/(Fclk * Fdata * X)



Atrenta Confidential © 2014 Atrenta Inc.

Typical Approaches and Their Limitations

SPYGLASS® FROM ATRENTA

- **Timing verification only checks Synchronous Logic.**
  - STA designed for synchronous modules/blocks
  - STA can report ALL async crossings, but can't differentiate between good/bad CDC's
  - Paths crossing clock domains are set to false paths and ignored.
  - Happens late in the design cycle at netlist level

- **Functional simulation very incomplete.**
  - Blackbox testing is too cumbersome and not exhaustive
  - Whitebox testing requires you to write assertions
  - Requires test benches for all the crossings
  - Is very reliant on luck (since simulation is not exhaustive), and will only detect a small subset of errors
  - Detected late in the design cycle

Clearly, Existing Techniques are Inadequate for CDC

Atrenta Confidential © 2014 Atrenta Inc.

Types of CDC Problems

SPYGLASS® FROM ATRENTA

- **Problem #1: Metastability**

  How do you ensure that you have isolated metastability across clock domains so it does not affect design functionality?

- **Problem #2: Convergence (correlation/coherency/re-convergence)**

  How do you ensure that a group of converging synchronized control signals are in sync at a particular clock cycle?

- **Problem #3: Functional problems (data loss, gray encoding)**

  How do you ensure that data from one clock domain is held long enough to be captured by the other clock domain? (Fast to slow clock data transfers)

- **Other CDC issues:**
  - Reset synchronization

    How do you ensure that an asynchronous set/reset is properly synchronized for a set of flops, and the reset is asynchronously asserted and synchronously de-asserted

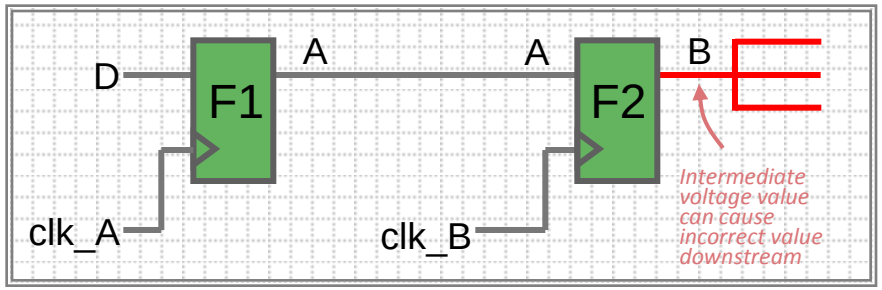
  - Constraint and exception validation

    How do you ensure that your clocks, set_case_analysis, quasi_static, waivers, Incomplete Formal Checks are accurate?

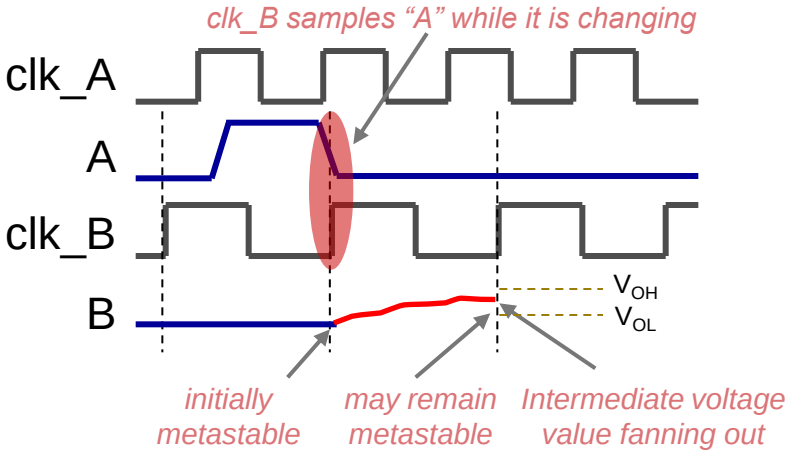
Atrenta Confidential © 2014 Atrenta Inc.

Problem #1: Metastability

Metastability problem



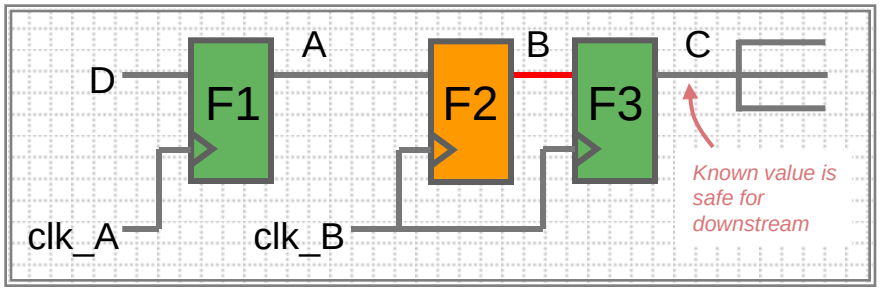
Output "B" becomes metastable whenever input "A" violates setup & hold time

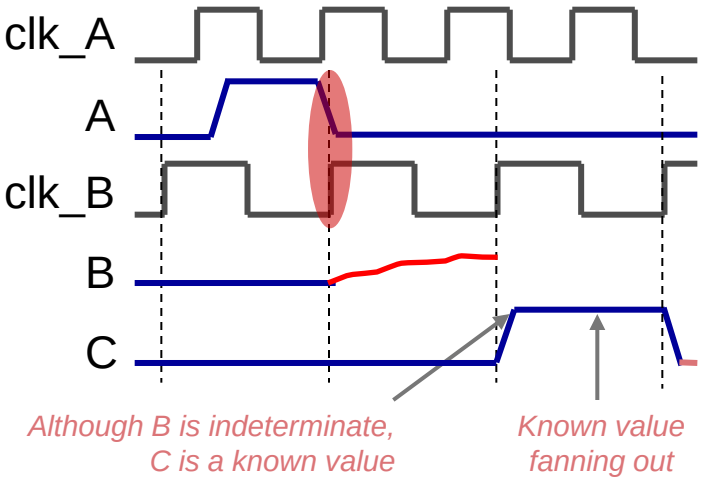


- **Why is this an issue?**
  - The output of flop F2 is unpredictable
  - This may lead to chip failures.
- **How can SpyGlass CDC Help?**
  - SpyGlass CDC identifies cases where synchronization is missing in the destination domain

Atrenta Confidential © 2014 Atrenta Inc.

Problem #1: Metastability





What designers typically do:

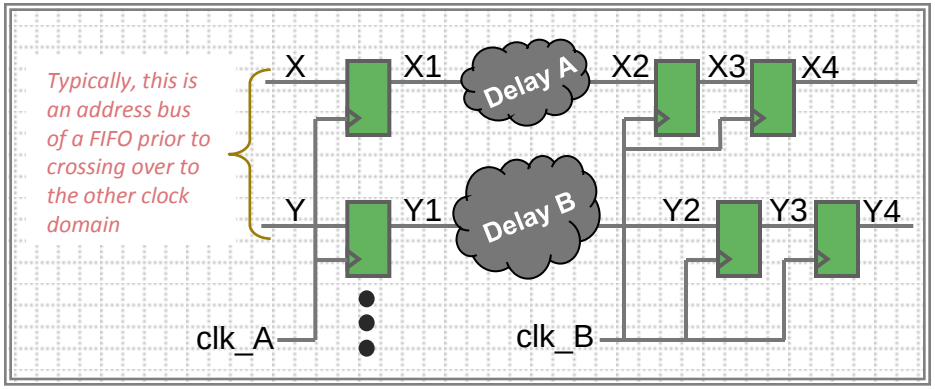
- Add additional flops in the destination domain to isolate meta and increase MTBF
- Add a mux recirculation flop in the destination domain
- Add a synchronization cell in the destination domain

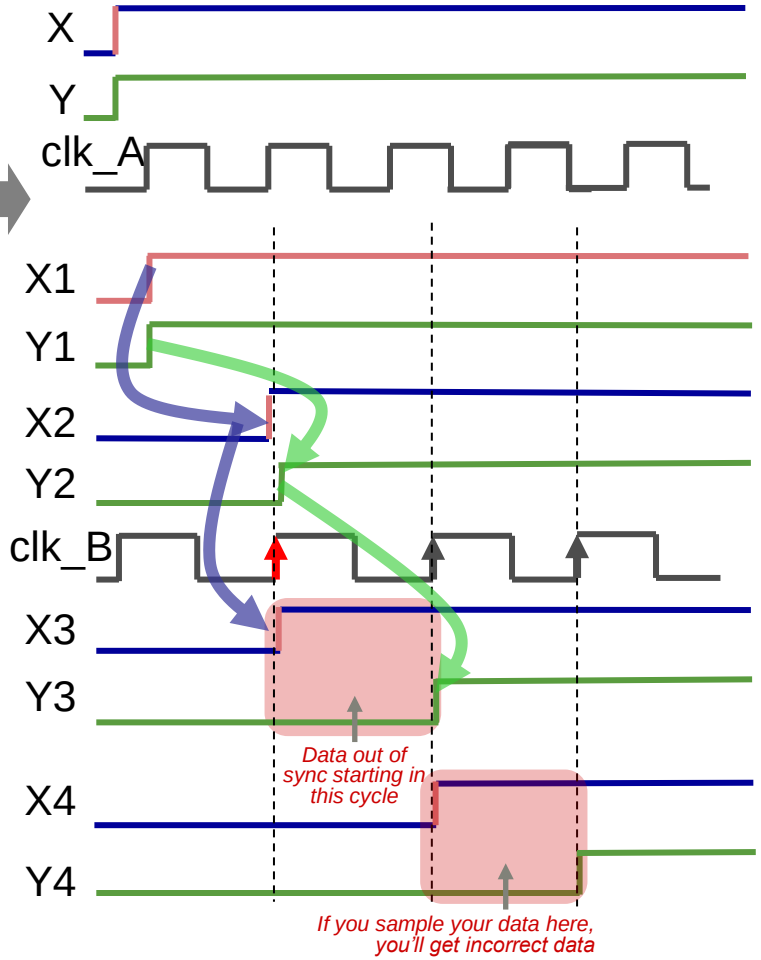
Note: Meta-stability can NEVER be eliminated

- In can only be isolated so MTBF can be increased
- Multi-Flop sync cells creates N clock cycles of Uncertainty
- Side Effect is data-coherency / re-convergence issues

Atrenta Confidential © 2014 Atrenta Inc.

Problem #2: Coherency of CDC Signals





Why is this an issue?

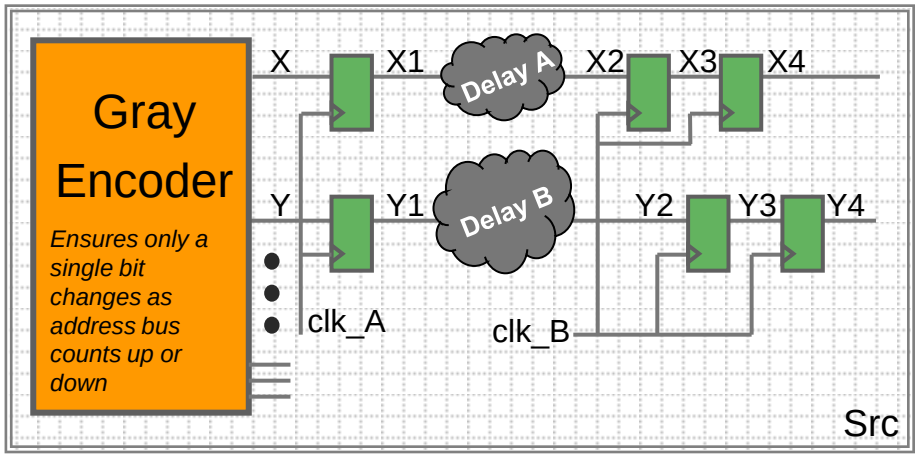
- **Meta-stability introduces latency and cycle uncertainty of N cycles in the receiving domain**
- **This may cause an invalid value being captured in the receiving domain. For example, in above picture clk_B domain captures an invalid state of 10**
- **High cause of chip failures**

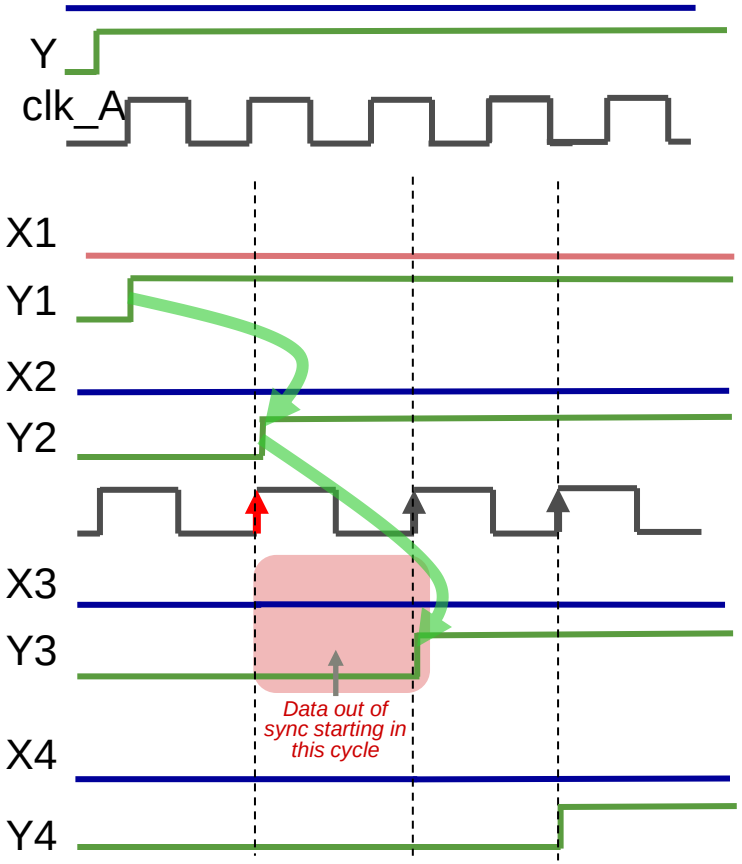
Src X:Y	00	11	11	11	11	11
Dest X4:Y4		00	00	10	11	

Atrenta Confidential © 2014 Atrenta Inc.

Problem #2: Coherency of CDC Signals

- **How do designers solve this problem:**
  - Gray encoding the signal, so it does not enter any invalid state
  - Verify signals can never change in the same destination clock cycle and waive the violation





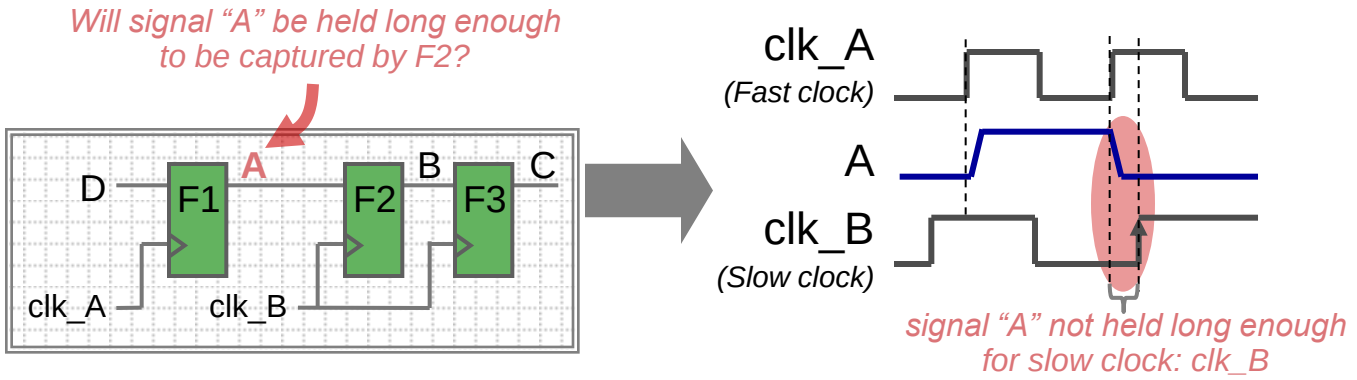
X:Y	00	01	01	01	01	01
Dest X4:Y4	00	00	00	01		

Atrenta Confidential © 2014 Atrenta Inc.

Problem #3: Data Hold

Data Hold Problem

(Signal crosses from a fast clock domain to a slow clock domain)

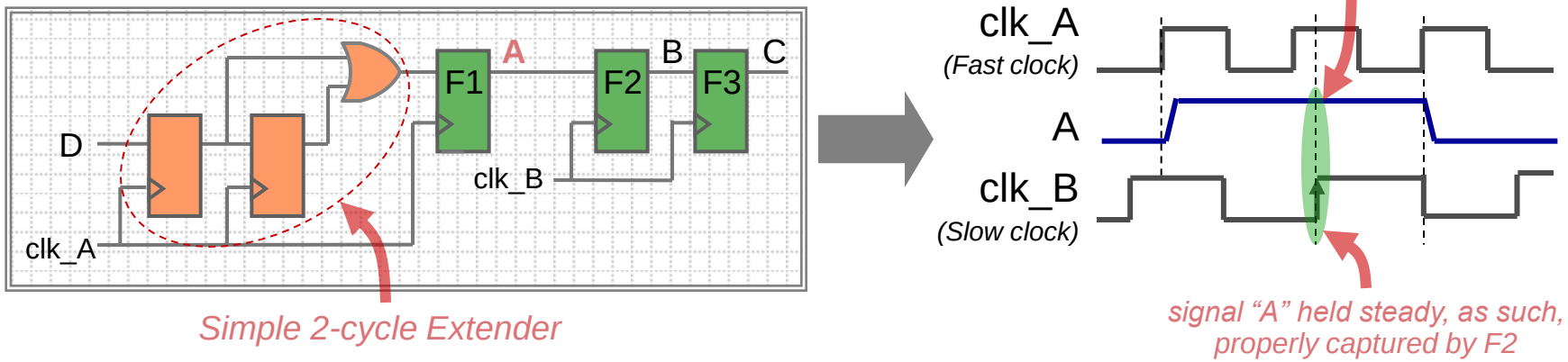


- **Why is this an issue?**
  - It's possible that the signal in the faster clock domain of F1 is not held long enough to be captured in the much slower clock domain of F2
  - If you do not fix this violation, data change may not be captured properly

- **How can SpyGlass CDC help?**
  - For every CDC crossing, a formal check is automatically carried out.
  - If the data crossing the domain crossing is not captured
  - In a DATA crossing if the Source Data is not Stable.
  - A waveform trace is generated

Atrenta Confidential © 2014 Atrenta Inc.

Problem #3: Data Hold



- **What designers typically do?**
- **Implement pulse extender or a state machine**
  - This ensures a good margin between the data being stable and the clock edge being active

Atrenta Confidential © 2014 Atrenta Inc.

Quiz

SPYGLASS® FROM ATRENTA

1. **What is the difference between synchronous clock domains and asynchronous clock domains?**

   Asynchronous clock domains have no guaranteed phase relation between clocks.

2. **Are synchronized clock domain crossings safe?**

   NO –They will ALL eventually FAIL. It is a matter of time. MTBF=1/(Fclk*Fdata*X). Goal is to maximize MTBF by putting in more sync flops.

3. **Name the two key problems caused by asynchronous clock domain crossings**

   METASTBILITY (output unpredictable) and data COHERENCY (correlated becomes uncorrelated)

4. **Do meta-stability problems depend on the frequency of the destination clock?**

   YES! The FASTER the CLOCK and DATA the SMALLER the MTBF

5. **Can static timing tool like Synopsys Prime Time detect asynchronous clock domain crossings?**

   YES they IDENTIFY CDC's. BUT, STA (PrimeTime) CAN NOT be used for clock domain crossing verification. They can identify CDC, but can't classify them as GOOD or BAD!

6. **Can clock tree synthesis (CTS) fix clock domain crossing issues?**

   NO –Clock Tree Synthesis is used to balance and optimize clock trees to insure synchronous transfers of DATA.

Atrenta Confidential © 2014 Atrenta Inc.

Agenda

An Introduction to CDC

The SpyGlass CDC solution

- CDC setup
- CDC setup check
- CDC structural verification
- CDC functional verification

Appendix

- Atrenta support information

Atrenta Confidential © 2014 Atrenta Inc.

Intro to SpyGlass CDC Issues, Goals, Rules

SPYGLASS® FROM ATRENTA

- **SpyGlass CDC is a set of rules that find issues related to:**
  - Safe and complete setup
  - Meta-stability (-rules Ac_sync/Ac_unsync)
  - Fast to slow clock crossing data loss (-rules Ac_datahold01a)
  - Convergence data coherency/correlation (–rules Ac_conv)
- **Rules are grouped together as goals**
  - For information management and user guidance
  - Always resolve rules from the TOP to Bottom. Waive or constrain if not an issue
- **Goals are progressive (e.g., cdc_setup_check is included in cdc_verify goal)**
  - **cdc/cdc_setup** =>SpyGlass finds your clocks, resets, clk mux, config registers, inputs
  - **cdc/cdc_setup_check** -> Is there anything missing or wrong in my setup?
  - **cdc/cdc_verify_struct** -> Structural CDC issues. Run often and resolve issues ASAP
  - **cdc/cdc_verify -> Signoff goal Including formal rules (Covered in advanced training)**
  - **cdc_validate, cdc_abstract, block_extract (Covered in SoC training)**
    - SoC verification or constraint extraction

Atrenta Confidential © 2014 Atrenta Inc.

Getting Started With SpyGlass CDC

- **Assumptions**
  - SpyGlass 5.3.0+ installed (Earlier version may not have some features)
  - If you have an IPKIT or CAD installed infrastructure then refer to that training
  - GuideWare 2.0 Goals or IP KIT goals (rule ordering supported)

- **Verify SpyGlass version:**
  - TYPE: spyglass –version

- **All SpyGlass jobs start with a project file (Next 2 slides)**
  - Project file defines: where to find source files, constraints, options, rules

- **SpyGlass can run in BATCH or GUI mode**
  - BATCH:
    - TYPE: spyglass –project training.prj –goals cdc/cdc_setup **–batch**
  - GUI:
    - TYPE: spyglass –project training.prj –goals cdc/cdc_setup

- **Typical use is to run analysis in batch mode then DEBUG the results using GUI**
  - It is easier to edit project file by hand than use GUI
  - GUI can interactively create constraints on-the-fly as necessary

Atrenta Confidential © 2014 Atrenta Inc.

TCL Project File Overview

- **Project file tell SG where to find RTL, Libs, Constraints, Run Options/Parameter, Rules, etc..: (Example on next page)**
  - RTL Source files or source files lists (Verilog/VHDL/netlists)
  - Constraints (SpyGlass specific or SDC)
    - Clocks, Resets, set_case_analysis, etc.
  - Run options (set_option / set_goal_option)
    - Top level module, System Verilog, library mappings, etc.
  - Rule specific (set_parameter)
    - Parameters: Combo logic in crossing's are good or bad
  - Goal specific (set_goal_option)
    - Parameters and options and enable/disable specific rules
    - Scenarios/modes where you run the same design but with different constraints

Atrenta Confidential © 2014 Atrenta Inc.

TCL Project File Example

```
#Training Project File
#This file supports all Standard TCL constructs in addition to SpyGlass specific constructs.
#Set the Top of the Design Tree
set_option top training
#Read in individual RTL Files (Verilog/VHDL/Netlists):
read_file -type hdl RTL/training.v
#Read in a file list from an location defined in an environmental file
###read_file -type sourcefile $env(SOURCE)/filelist.f
#Read in a Design Constraints File that contains clocks, resets, set_case_anlysis, etc..
read_file -type sgdc SGDC/training.sgdc
#Define a waiver file to put EXCEPTIONS INTO and a default waiver file.
read_file -type waiver training.swl
set_option default_waiver_file training.swl
#Set a global option that all Verilog will be System Verilog Compliant.
set_option enableSV yes
#Where do we find our rule sets:
current_methodology $env(SPYGLASS_HOME)/GuideWare2.0/block/rtl_handoff
#Define goal specific parameters and options
current_goal cdc/cdc_verify
set_parameter ac_sync_debug yes #Turns on extra debuggin during cdc_verify goal
set_parameter enable_handshake no #Turn off Handshake Sync Scheme
set_goal_option XXXX # Only for this goal
#Define a scenario that uses different SGDC file "-goals cdc_verify@mode1"
current_goal cdc/cdc_verify -scenario mode1
read_file -type sgdc mode1.sgdc
```

Atrenta Confidential © 2014 Atrenta Inc.

Goal: cdc_setup
Defining Initial Clock /Reset Conditions

- **SG-CDC needs clocks and resets in order to do CDC analysis**
  - Specify Clocks/Resets in a **S**py**G**lass **D**esign **C**onstraint file (SGDC)
  - Without Clocks/Resets, SpyGlass will not be able to perform CDC analysis
    - SpyGlass will complain loudly about missing Clocks/Resets/SCA
  - An incorrectly specified constraint can lead to false violations, unnecessary debugging time and frustration.

- **SG can automatically infer Clocks, Resets, Set_case_analysis, Quasi_static.**

- **Optionally remove complex clock logic from the analysis.**
  - It is easier to specify clocks on boundaries of a Clock Block instead of trying to figure every set_case_analysis on every mux to constrain specific clocks.

- **If you already know the clocks and resets specify them**

- **If you have a Synopsys Design Constraint (SDC) file**
  - This can be used directly or translated to SGDC
  - Only contains Clocks and SCA. Still need to provide resets.

Atrenta Confidential © 2014 Atrenta Inc.

SpyGlass Design Constraint File

- **SGDC = SpyGlass Design Constraints**
  - Defines constraints similar to what an SDC file does for DC/STA.
    - SDC is not sufficient since it doesn't describe resets or CDC specific data
  - Example SGDC constraints
    - Clock –name <port|net> -domain <unique domain name> -period <10> -edge <0 5>
    - Reset <–async|-sync> –name <reset port|net name>
    - set_case_analysis
      - The *set_case_analysis* constraint specifies the case analysis conditions. It uses the following syntax:

        set_case_analysis -name {<name>} -value <value>

        The value set on the signal specified with the set_case_analysis keyword is automatically propagated through the design
    - quasi_static –name <port|reg|net name>
      - There are some primary inputs, flip-flops, or nets present in the design, which may change briefly at the start but assume a static value of 0 or 1 for the rest of the circuit operation.
      - You can specify the primary inputs, flip-flops, or nets as static signals using the *quasi_static* constraint to skip the verification of paths that involve such signals
    - Design cell constraints
      - sync_cell –name <module names> -other_options
      - reset_synchronizer –reset <reset port> -clock <clock> -name <reset output name>

- **Location of SDC file for sdc2sgdc conversion**
  - sdc_data –file my.sdc

Atrenta Confidential © 2014 Atrenta Inc.

SGDC File Example

```
//Scope the constraints to a specific module.
current_design training
//clocks and set_case_analysis come from SDC file
sdc_data -file …../my_file.sdc

#DEFINE MY CLOCKS
clock -name clkA #Incomplete clock constraint, missing -domain -period and -edge
clock -name clkB -domain SysClck -period 10 -edge 0 5

#DEFINE MY RESETS
reset -name reset_n #Incomplete reset constraint, missing active value
reset -name reset_n -value 0 #Complete reset Constraint

#DEFINE MY INPUT CLOCK RELATIONSHIPS
abstract_port -ports reset -clocks Vclk

#DEFINE MY STATIC SIGNALS
Quasi_static -name "training.u2.config_reg[1]"
```

Atrenta Confidential © 2014 Atrenta Inc.

Clock Generation Logic Recommendations

- **Recommendation: Blackbox the clock generation logic if you don't have all the SCA.**
  - IP that contains complex clock generation logic and requires lot of set_case_analysis to configure properly, are best blackboxed
  - Clocks and resets can be defined on outputs of clock generation logic or module.
  - Otherwise time is wasted looking at special crossings inside the clock generation logic and trying to configure it in proper functional mode.

- **Some clocks are harmonically related, though different frequencies.**
  - Put them in the same domain using –domain on clock constraint.

- **Could have same frequencies but different domains.**
  - Put them in different domains with –domain.

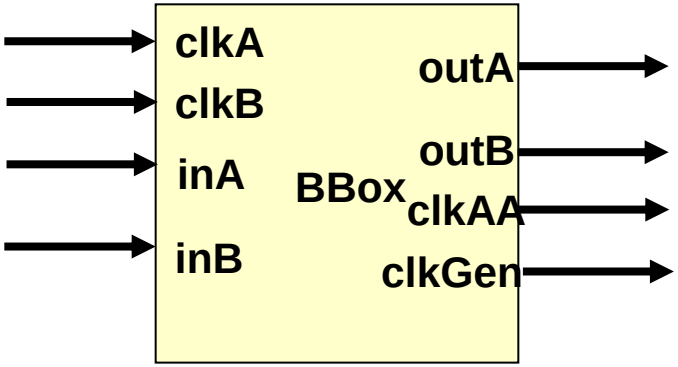
- **Exceptions:**
  - If SDC is provided that configures clock generation logic
    - If you blackbox the clock generation logic you will get fatal errors on objects that don't exist in design due to blackboxes
    - Since SDC is provided then most configuration issues have already been defined

- **Tricks:**
  - CTS/CRC (clock tree synthesis/clock root cells) could be blackboxed so clocks get identified at these cells. These are usually unique domains so they make good starting points for clocks

Atrenta Confidential © 2014 Atrenta Inc.

Handling BlackBoxes (BBox)

- **There are two types of Bbox's.**
  - GOOD BBox – These can be ignored
    - Liberty Models with related_pin info. (Warning)
    - BBox with single clock will assume all I/O on single clock domain.
  - BAD BBoxs – These produce "noise" in CDC analysis
    - Bbox with multiple clocks
- **Parameters control the level of analysis and reporting during CDC analysis**
  - cdc_reduce_pessimism=bbox (This will filter all crossings to/from Bbox. Caution)
- **2 Options for handeling Bbox's.**
  - 1) Treat it as a constraint model using "signal_in_domain"
  - 2) Create an Abstract Model of block and use SoC Validation (Refer to SoC Training)



SGDC file

```
current_design BBox
#Define port  domain information
signal_in_domain –name Bbox –signal inA –clock clkA
signal_in_domain –name Bbox –signal outA –clock clkA
signal_in_domain –name Bbox –signal inB –clock clkB
signal_in_domain –name Bbox –signal outB–clock clkB
#Feedthrough for clock port
assume_path –name Bbox –input clkA –output clkAA
#Internally Generated Clock
Clock –name clkGen –domain new –period 10 –edge 0 5
```

Atrenta Confidential © 2014 Atrenta Inc.

Running SpyGlass for the First Time

- **Assumptions: You already have a project file**

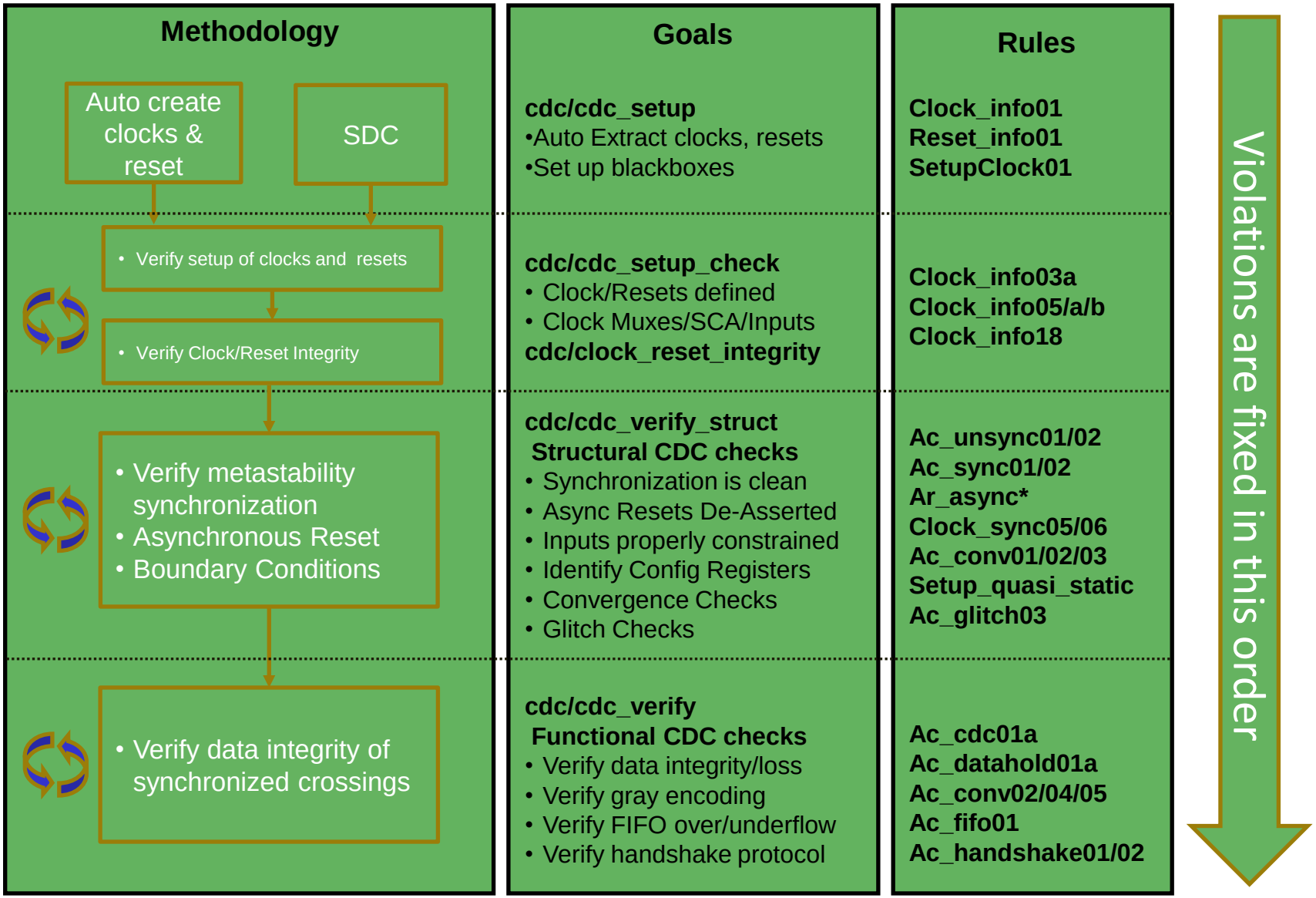
- **First time you run SpyGlass it will analyze, elaborate and synthesize the design**
  - Then it traces backwards from flops clocks and resets to:
    - Primary Inputs (primary clocks)
    - Blackbox (blackbox clocks)
    - Flops (generated clocks/resets)
  - DEFINITE CLOCK = If it is a direct path, then it is a known clock
  - PROBABLE CLOCK = IF it goes through combo-logic
    - Then one of the inputs could be gating/enable logic
  - You have to review all clocks/reset to:
    - Confirm correctness, Add domain Info, add period/edge information, active value

- **Command line to find clocks:**
  - `spyglass -project training.prj -batch -goals cdc/cdc_setup`
  - NOTE! If you forget the name of the goal, then abbreviate:
    - `spyglass -project training.prj -batch -goals cdc`
      - » It will return all goals that have the string “cdc” in them and you can copy/paste

- **Clocks, resets and other setup files will be found in:**
  - Project name/top level name/GOAL NAME/spyglass_reports/clock_reset
    i.e., training/training/cdc/cdc_setup/spyglass_reports/clock_reset

Atrenta Confidential © 2014 Atrenta Inc.

SpyGlass CDC Flow



Atrenta Confidential © 2014 Atrenta Inc.

Setup for SpyGlass CDC Verification

Purpose

- Using SpyGlass, define clocks and resets as a preparation step for setup check and clock domain crossing verification

Pre-requisites

- Zero errors and zero warning during the designread

Command

- %shell> spyglass -project lab-CDC_proc.prj -goals cdc/cdc_setup -batch

Output

- SGDC files with clock/ reset information. The following are the files generated:
  - autoclocks.sgdc
  - generated_clocks.sgdc
  - autoresets.sgdc
  - generated_resets.sgdc

Atrenta Confidential © 2014 Atrenta Inc.

After Running CDC/cdc_setup Goal

- **AIPK copies the clock/resets automatically to your block.sgdc file.**

- **Examine the files autoclocks.sgdc/generated_clocks.sgdc**
  - **training/training/cdc/cdc_setup/spyglass_reports/clock_reset/autoclocks.sgdc**
  - Are these real clocks or are some control signals? (testmode, clock enable)
  - Clocks can be added to your own SGDC file or this file copied to your local area
  - Confirm the –domain is correct. –domain is how you group synchronous clocks
  - Control signals should get a `set_case_analysis` on them

- **Examine the file autoresets.sgdc**
  - **training/training/cdc/cdc_setup/spyglass_reports/clock_reset/autoresets.sgdc**
  - Are these your real resets or are some control signals?
  - Do they have the right –value <acative reset value=1/0> 'X=Can't determine'
  - resets can be added to your own SGDC file or this file copied to your local area
  - Control signals should get a `set_case_analysis` on them

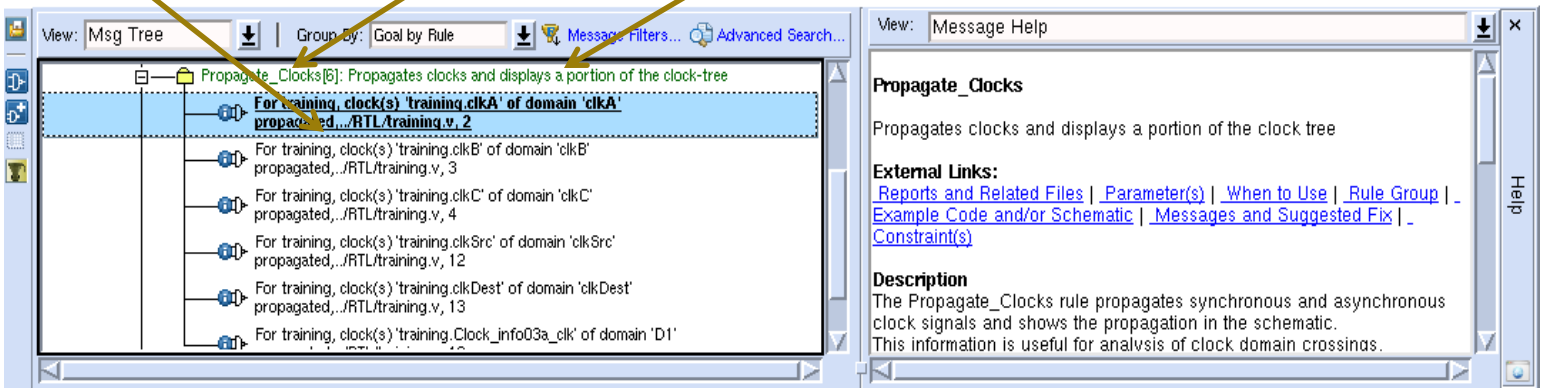
- **There are some other files that should also be looked at and used if necessary**
  - Generated_clocks.sgdc – Any clock dividers will create generated clocks
    - These should get used in addition to autoclocks.sgdc
  - Auto_set_case_analysis.sgdc
    - Clocks that go through muxes need to have a `set_case_analysis`.
    - This file identifies them so you can set them up-front before they cause problems later

Atrenta Confidential © 2014 Atrenta Inc.

Using GUI to Help Examine Clocks/Resets

- **Many times you may be unfamiliar with the clocks or resets of the block you are running. Using the GUI can save a lot of time tracing signals through hierarchy**
- **Open the GUI:**
  - spyglass –project training.prj –goals cdc/cdc_setup
    - NOTE! If you leave off the –goals, you can choose the goal when the GUI opens
- **Configure the group by: Goal by rule**
  - This is important, as lots of rules violate, so you will know where to start.

- **RULES are grouped in FOLDERS with "Rule Name" and short description**
  - Click on FOLDER or RULE to get HELP
- **Individual rules have details –**
  - Read messages carefully and they will guide you to the issue



Atrenta Confidential © 2014 Atrenta Inc.

Rules that identify Likely Clock & Reset Signals

- **Clock_info01 – reports likely clock signals**
- **Reset_info01– reports likely reset signals**
- **Goal: cdc/cdc_setup**
- **Why: This automatically finds clocks and resets for you**
  - If you already know your clocks, then you don't need this rule
  - It always generates a complete set of clocks. It is not incremental
  - It puts the clocks in the …./spyglass_reports/clock_reset/autoclocks.sgdc
- **Schematic info – Shows likely clock to the first flop it finds**
- **What to do? Determine if this is a REAL clock or control signal**
  - Is it in the correct domain?
- **Issues: The autoclock.sgdc file is incomplete and assumes all clocks are async**
  - The clock constraint automatically puts every clock in a unique –domain
  - The –domain argument is just text that describes the domain
    - A good practice is to change it to the name of the clock
  - It is missing –period X.XX and –edge 0 Y.YY information
    - If you don't add this information, SpyGlass will trigger the Ac_clockperiod01 rules

Atrenta Confidential © 2014 Atrenta Inc.

Agenda

- **An Introduction to CDC**
- **The SpyGlass CDC solution**
  - CDC setup
  - CDC setup check
  - CDC structural verification
  - CDC functional verification
- **Appendix**
  - Atrenta support information

Atrenta Confidential © 2014 Atrenta Inc.

Clock / Reset Setup Check

SPYGLASS® FROM ATRENTA

design read → cdc_setup → cdc_setup_check → cdc_verify_struct → cdc_verify

Purpose

- The purpose of this step is to perform constraints sanity checks and detect any conflicting/missing constraints. For correct CDC analysis, it is very important to define the clocks and reset correctly in the constraint file
- If clocks are properly defined, this goal will also generate input domain constraints

Pre-requisites

- Zero errors and zero warning in the previous goal cdc/cdc_setup
  - Clocks, resets, and mode settings properly set
  - All rules from cdc_setup are included in this goal too

Command

- *%shell> spyglass -project lab-CDC_proc.prj -goals cdc/cdc_setup_check -batch*

Output

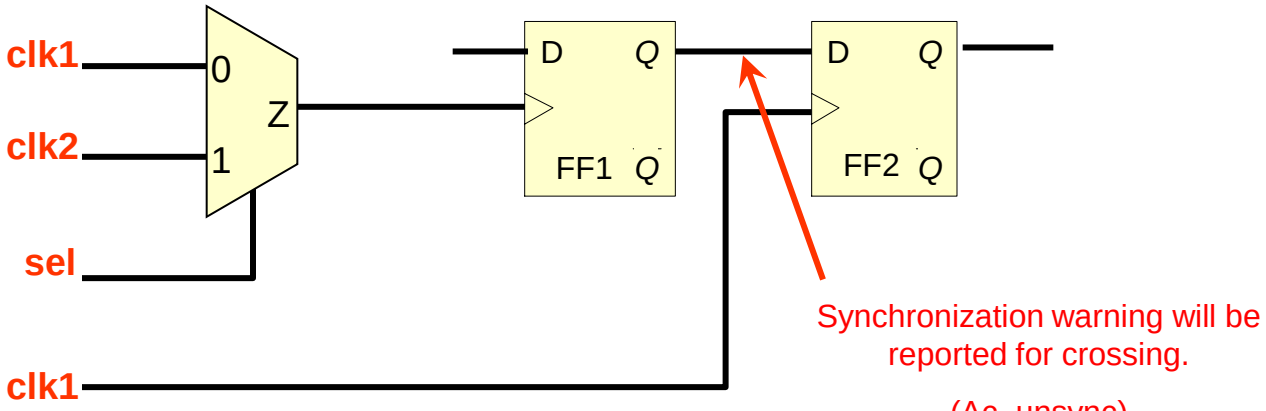
- List of possible setup problems reported as rule violations. The CDC-Setup-check template runs the following rules:
  - Clock_info03a - reports flops with unconstrained clocks
  - Clock_info03b - reports flops with data pin tied to constant
  - Clock_info05 - reports clock signals converging on muxes

Atrenta Confidential © 2014 Atrenta Inc.

Dealing With Clock Signals Converging on Muxes

- **With no case analysis**
  - FF1 will assume a merged domain which is different than clk1 or clk2
    - Without constraining the clock-mux, domain crossing will be reported because source and destination flops can receive different clocks
    - With case analysis the source and dest clock are the same so there is no CDC
      - » set_case_analysis -name sel -value 0
  - Clock_info05 identifies pure MUX's
  - Clock_info05a identifes clocks merging on logic. (Glitch Free Clock Mux).

- **Recommended method, use**
  - It is important for you to apply a case analysis constraint on the select pin `sel` of the mux



Atrenta Confidential © 2014 Atrenta Inc.

Quiz

1. **Name two important constraints required to do CDC analysis**
   1) Clock 2) reset 3) set_case_analysis

2. **Name three sources of finding all the clocks and resets to be constrained for CDC analysis**
   1) Design Spec 2) Infer from RTL 3) SDC file

3. **Should clock-generation/manipulation logic be part of CDC analysis?**
   Generally NOT, unless you fully understand how to constrain.

4. **How do you exclude clock generation/manipulation logic from CDC analysis?**
   -stop module (Bbox's a module)

5. **Name one constraint to create association between CLOCK and OUTPUT pins of a blackbox to do CDC analysis**
   signal_in_domain –name MY_PORT -clock MY_CLOCK

6. **Name a rule to detect unconstrained clock nets**
   Clock_info03a – Unconstrained clock Nets

7. **How do you resolve multiple clocks converging on a mux?**
   Set_case_analysis or define clock on output of mux

Atrenta Confidential © 2014 Atrenta Inc.

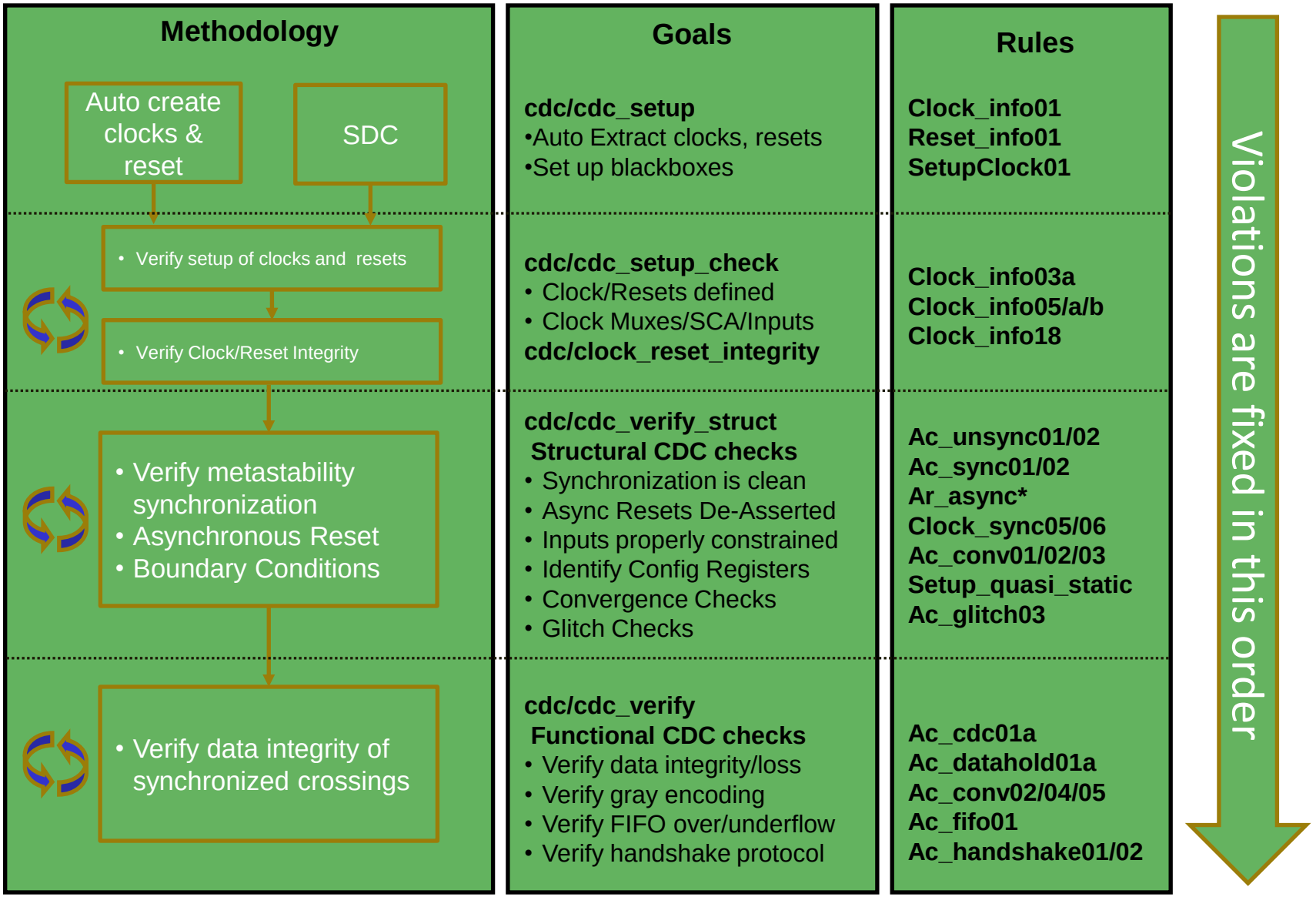
Hands-on Exercise Lab 1 and Lab 2

- **Hands-on exercise objectives:**
  - Lab1 : CDC setup Lab
    - ✓ Identify the clocks and resets in the design
    - ✓ Resolve blackboxes in clock tree path and crossing path
    - ✓ Perform reset setup and IO setup
  - Lab2: CDC setup check lab
    - ✓ Verify clocks and reset setup
    - ✓ Ensure all flops are receiving clocks

- **Duration:** 60 minutes

Atrenta Confidential © 2014 Atrenta Inc.

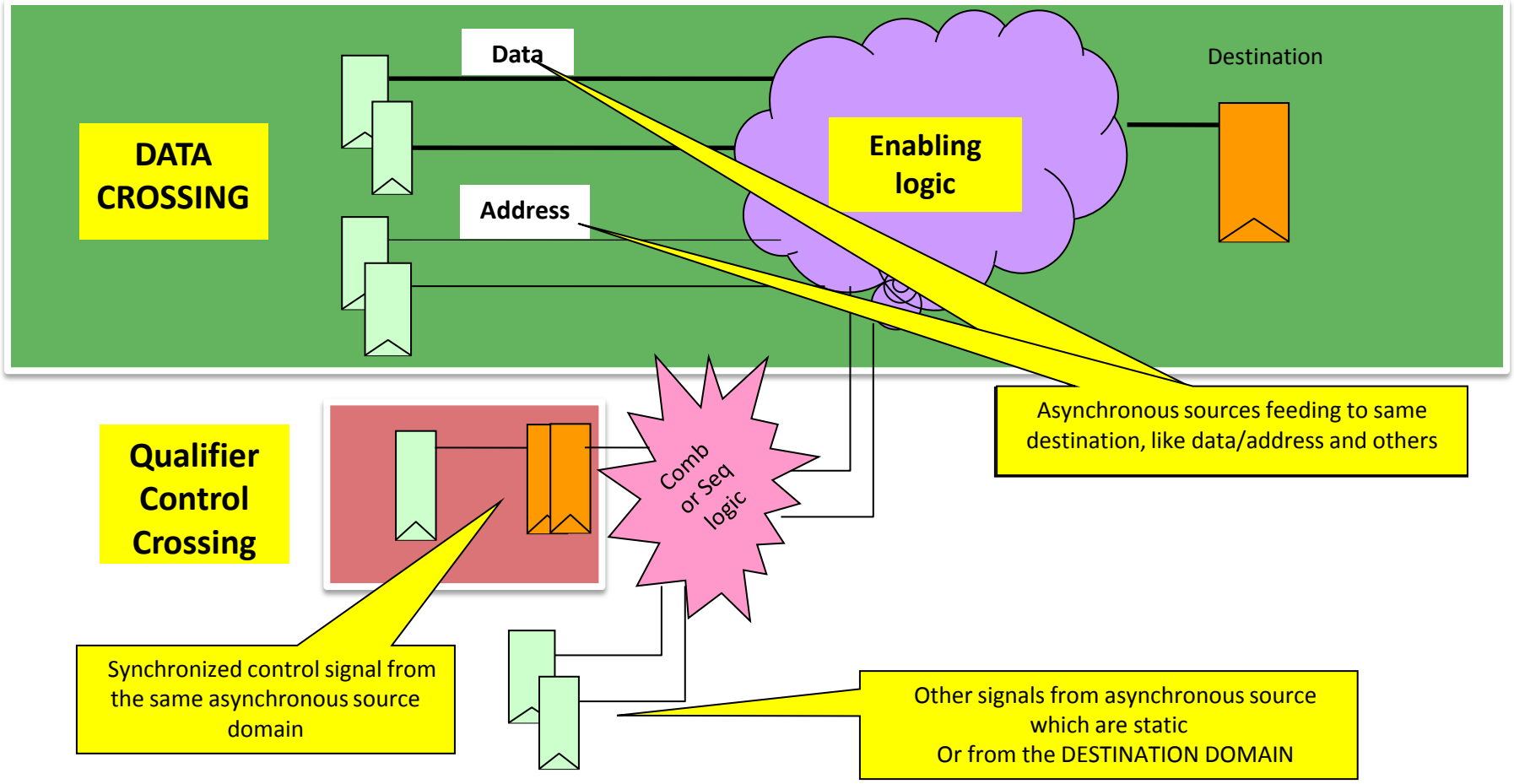
SpyGlass CDC Flow – cdc_verify_struct



Atrenta Confidential © 2014 Atrenta Inc.

Protocol Independent CDC Recognition

- Following picture represents ALL CDC's. (Fifo, Handshake, Mux, Enable, etc)
- SpyGlass FIRST identifies all qualifiers (multi-flop) crossings.
- Then for non-multi-flop crossings, SpyGlass looks for qualifier in fanin of dest. flop



Atrenta Confidential © 2014 Atrenta Inc.

CDC Rule Overview

- **SpyGlass sees all crossings as only 2 types of clock domain crossing's (CDC's)**
  - **Control crossings**: These are **muti-flop** or **sync_cell** or **qualifier constraint** crossings
    - Control crossings move control signal across a clock domain
    - Control crossings qualify data crossing
      - » Control crossings synchronizer is synonymous with **qualifier**
  - **Data crossings**: Can be scalar or vector signals
    - They are always qualified by a single control crossing.
    - A good data crossing will always contain a good control crossing
  - NOTE! If the control crossing is marked as unsynchronized, then the data crossing will also be unsync'd

- **Multi-flop -> Minimum of two back-to-back flops**
  - Constraint ***num_flops*** can configure SpyGlass to require more than two flops
  - Disable Multi-Flop and require ONLY sync_cell.
    - **set_parameter disable_multi_flop yes**

- **Sync_cell -> Module constraint to define a control synchronizer**
  - Defined on a single flop or library cell.
  - Many options to define frequency/periods where it can be applied
  - Replaces parameters (synchronize_data_cell (Vector) and synchronize_cell (scalar) synchronizers
  - Don't use for large block like a FIFO that contains both source and destination. Not safe!

- **Qualifier -> Constraint that can be applied anywhere in the design including inputs to define a control synchronizer**

Atrenta Confidential © 2014 Atrenta Inc.

Goal: cdc/cdc_verify_struct

Purpose

- Discover structural issues in clock domain crossings

Pre-requisites

- Zero errors and zero warning in the previous goal cdc/cdc_setup_check
  - Clocks, resets, and mode settings properly set
  - All rules from cdc_setup_check are included in this goal too

Command

- *%shell> spyglass -project lab-CDC_proc.prj -goals cdc/cdc_verify_struct -batch*

Output

- *List of possible clocking problems reported as rule violations*
- *Debug each violation in the rule order, iterate running SpyGlass*

Atrenta Confidential © 2014 Atrenta Inc.

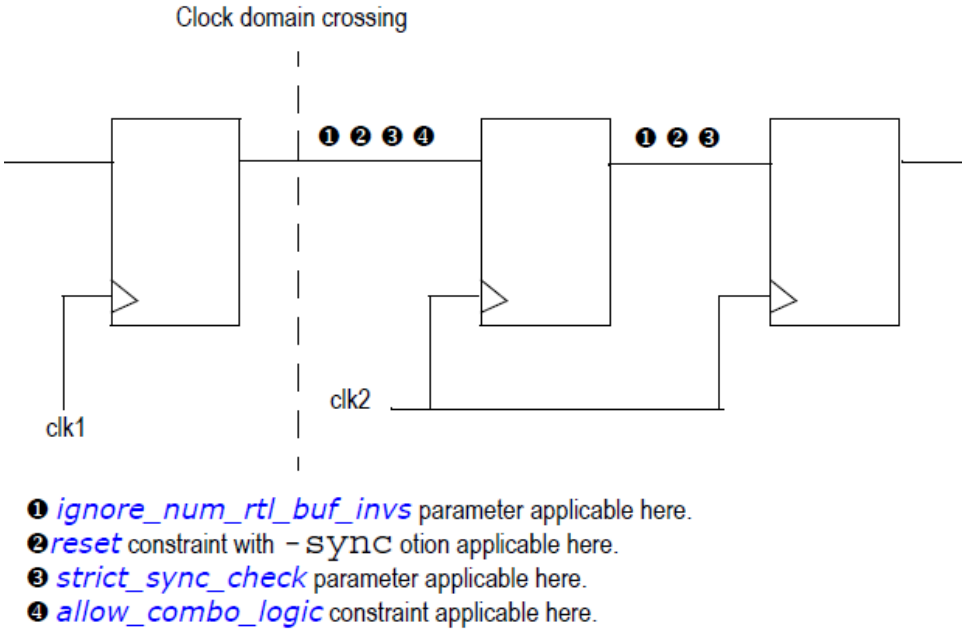
Examples of SpyGlass Recognized Synchronizers

- **The following slides show some of what SpyGlass automatically recognizes**
- **Also shows where some parameters can be applied to configure what is good and bad**
- **This is not an exhaustive list and new schemes get added with each release**
- **All sync schemes can be disabled if not allowed**
- **GuideWare2.0 has pre-selected the most robust and safest synchronizers and parameters.**
  - You should review to confirm!

Atrenta Confidential © 2014 Atrenta Inc.

Conventional Multi-Flop Scheme for Control Crossings

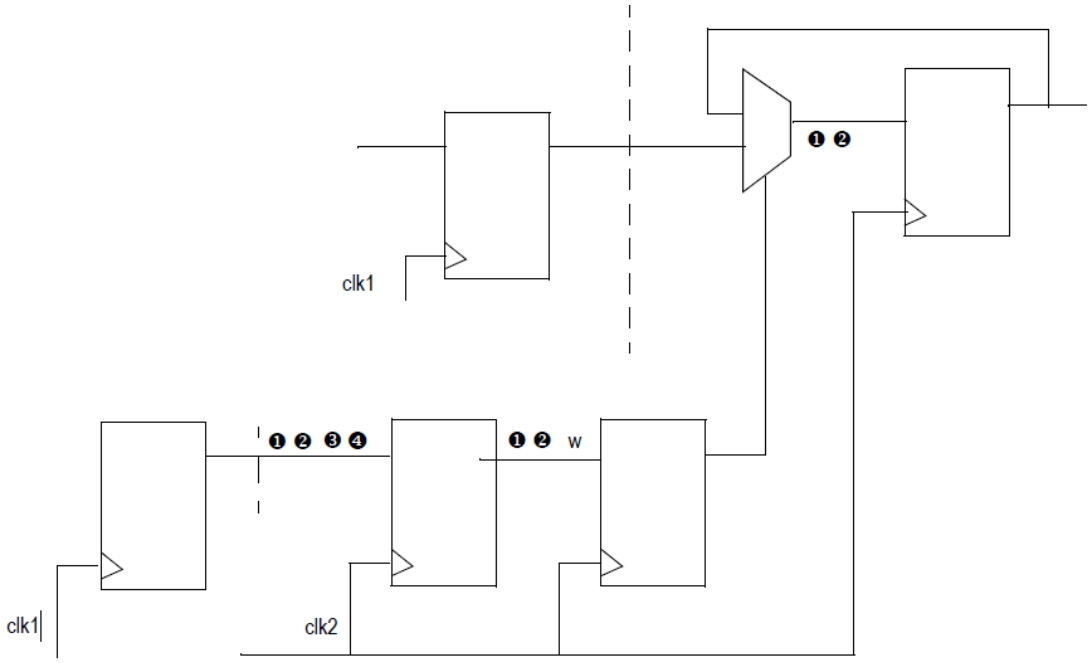
- **Control crossings can be:**
  - RTL multi-flops (auto detected)
  - Liberty cells (require sync_cell constraint)
  - Single flops (require sync_cell constraint)
- **All data crossings contain at least:**
  - One control crossing



Atrenta Confidential © 2014 Atrenta Inc.

Recirculation Mux Sync for DATA Crossings

- Automatically recognized
- set_parameter enable_mux_sync all
- NOTE! It contains a control crossing

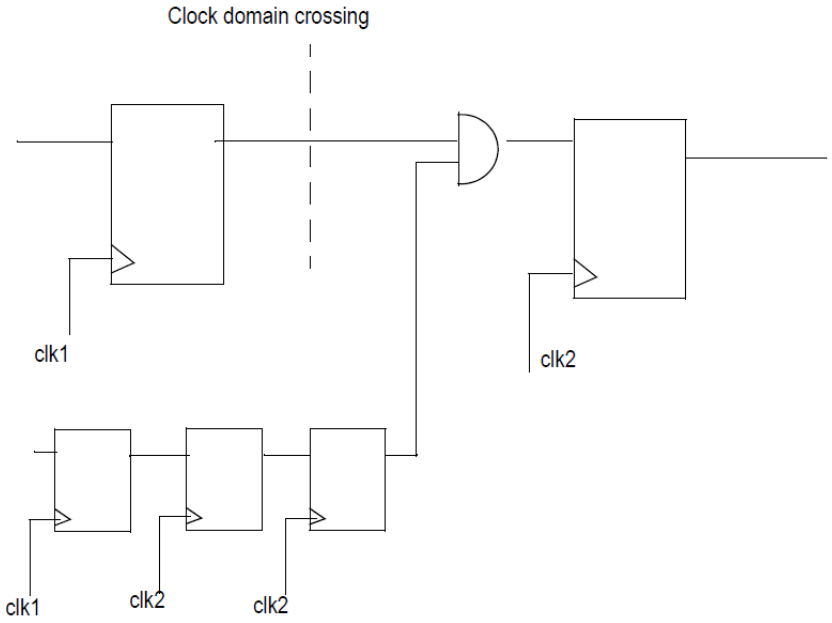


❶ *ignore_num_rtl_buf_invs* parameter applicable here.
❷ *sync_reset* parameter applicable here.
❸ *strict_sync_check* parameter applicable here
❹ *allow_combo_logic* constraint applicable here

Atrenta Confidential © 2014 Atrenta Inc.

AND Gate Sync Scheme for Data

- set_parameter enable_and_sync yes
- AND gate can be AND/OR/NAND/NOR
- XOR not allowed or complex AOI, unless AND gate does gating
- NOTE! It contains a control crossing



Atrenta Confidential © 2014 Atrenta Inc.

SpyGlass CDC Synchronization Schemes

- **SpyGlass CDC solution reports a clock domain crossing as synchronized based on the following different clock domain crossing synchronization schemes:**
  1. *Conventional multi-flop synchronization scheme (Control) – num_flops*
  2. *Synchronizing cell synchronization scheme (Control) – sync_cell constraint*
  3. *Delay signals synchronization scheme (Control or Data)*
  4. *Synchronized enable synchronization scheme (Data)*
  5. *Recirculation MUX synchronization scheme (Data)*
  6. *AND gate synchronization scheme (Data)*
  7. *MUX-select sync (without Recirculation) Synchronization scheme (Data)*
  8. *FIFO synchronization scheme (Data)*
  9. *Glitch protection cell synchronization scheme (Data)*
  10. *Clock-gating cell synchronization scheme (Data)*
  11. *Qualifier synchronization scheme (Data)*
  12. *Handshake synchronization scheme (Data For Request and Control for Acknowledge)*
  13. *MUX lock synchronization scheme (Data)*
  14. *Pattern synchronization scheme (Data)*

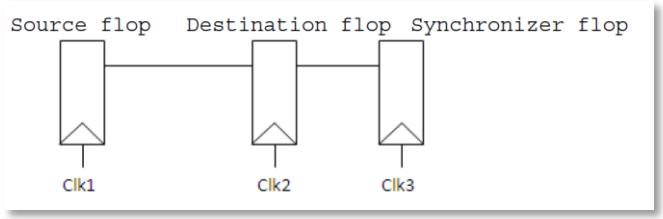
RED is for Control Crossings
Black is for Data Crossings

Atrenta Confidential © 2014 Atrenta Inc.

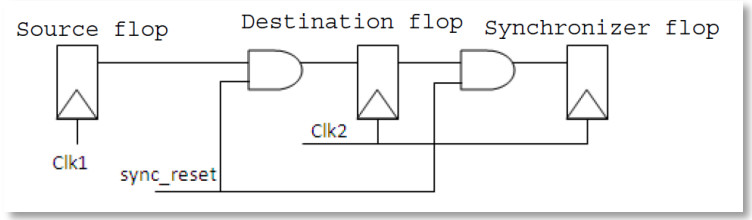
Some Reasons for Being Unsync'd (Ac_unsync)

SPYGLASS® FROM ATRENTA

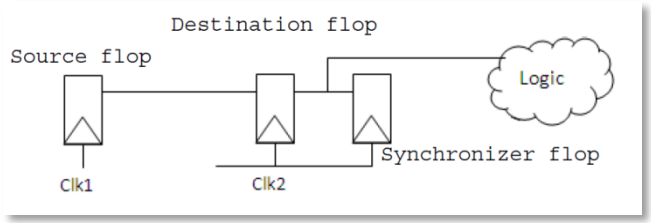
Clock domains of destination instance and synchronizer flop do not match



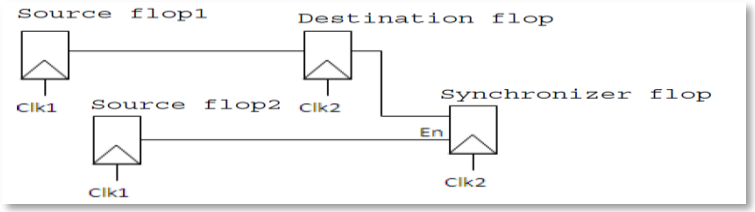
Sync reset used in multi-flop synchronizer



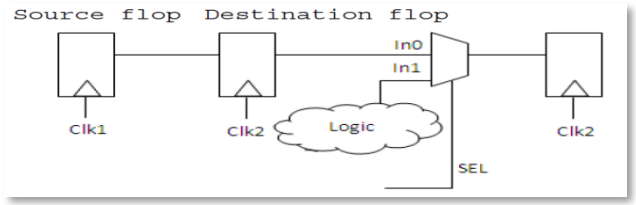
Destination instance is driving multiple paths



Synchronizer flop is the destination flop for another crossing

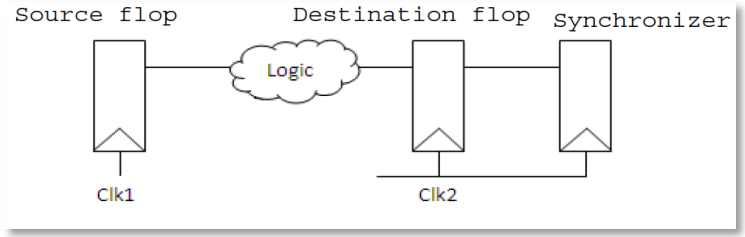


strict_sync_check failed for path between synchronizer flops



Combinational logic used between crossing is ALLOWED

Ac_glitch03 will identify these.



Atrenta Confidential © 2014 Atrenta Inc.

Unsynchronized Types (Not Exhaustive List)

- **You will spend most of your time looking at unsynchronized crossings in the Ac_unsync rule**
- **Every crossing has a reason for being unsynchronized**
  - ALWAYS READ THE REASON!!! This will save you a lot of time
- **Current list of unsynchronized crossings:**
  1. Sync reset used in multi-flop synchronizer (Control)
  2. Combinational logic used between crossing (Control) (OK->Ac_glitch03 reports)
  3. Destination instance is driving multiple paths (Control)
  4. Sources from different domains converging (Control)

     set_parameter cdc_reduce_pessimism ignore_multi_domain
  5. Synchronizer flop is the destination flop for another crossing (Control)
  6. **No valid synchronizer (Control and/or Data)**
  7. Merges with inferred qualifier at invalid gate (Data)
  8. Merges with inferred qualifier from invalid domain (Data)
  9. Clock phase difference between destination instance and synchronizer flop (Control)
  10. Clock domains of destination instance and synchronizer flop do not match (Control)
  11. Number of inverters/buffers between sync flops exceeds limit (Control)

Atrenta Confidential © 2014 Atrenta Inc.

Using SpyGlass to Resolve Unsync'd

- **Every single flop will get reported by SpyGlass as one of the following:**
  - Synchronized
  - Unsynchronized
  - No analysis due to stuck clock or data (Clock_info03b or Clock_info03c)

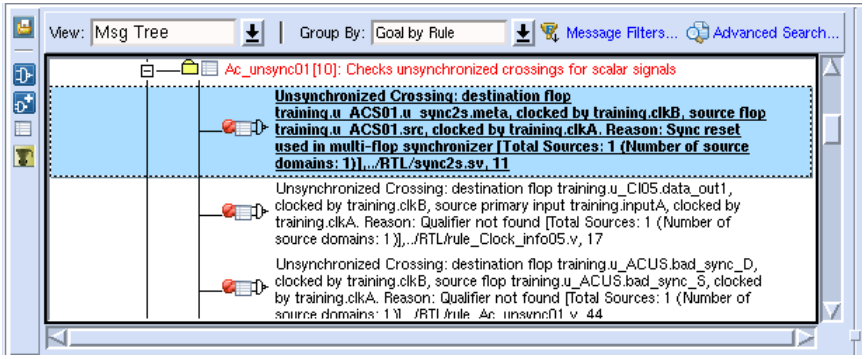
- **For every crossing, SpyGlass will report the REASON it is unsynchronized**
  - The REASON is the first place to look to verify a crossing or solve an unsync'd crossing
  - Some crossings could have multiple reasons. Only one reason is given at a time
    - For same destination flop there could be both sync'd and unsync'd sources
    - **Do not make a determination that it is good based on only looking at one source**

- **SpyGlass will shows CDC's in 2 ways**
  - Message tree/Moresimple.rpt
    - Limitation: Only shows one source/clk, dest/clock, reason
  - Primary spreadsheet
    - Similar limitation as message tree, but gives additional information of # sources, # domains
  - Secondary spreadsheet
    - Gives all information about every source in a crossing

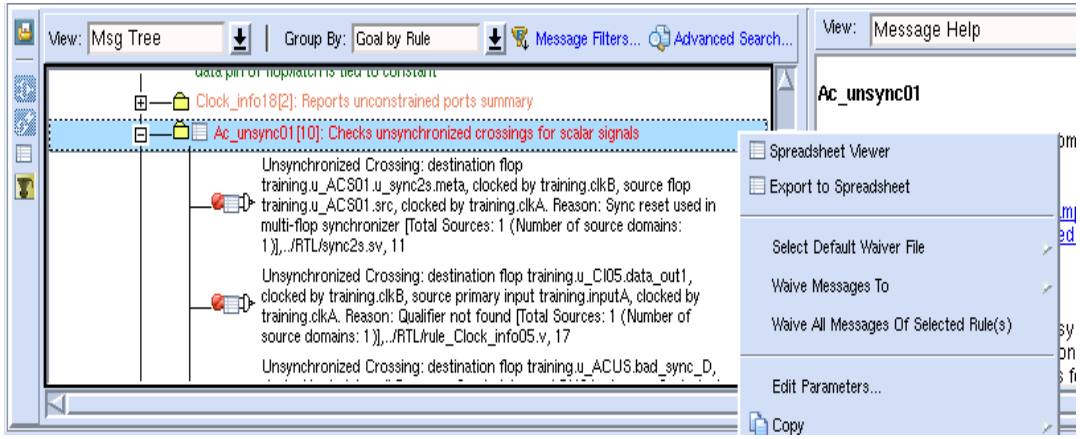
Atrenta Confidential © 2014 Atrenta Inc.

Debugging Unsynchronized Crossings

- **Don’t try to read the message tree**
- **This isn’t Lint!**
- **Use spreadsheet to view!**
- **It lists the IMPORTANT information**



- **One line/destination (synchronizer)**
  - Source
  - Source clock(s)
  - Destination
  - Destination clock(s)
  - **REASON**
  - **Total sources**
  - Total source domains
  - Source instance
  - Destination instance
  - Crossing module
  - Waived (don’t use)



- ***NOTE! IF there is more than one source, you MUST push into the detailed spreadsheet***

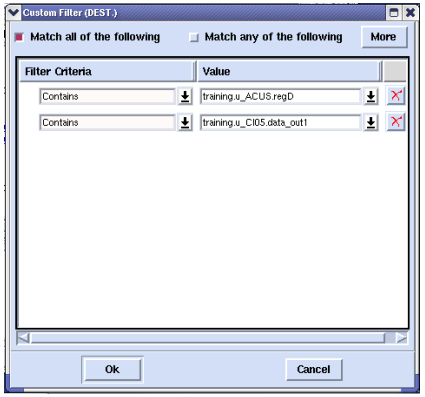
Atrenta Confidential © 2014 Atrenta Inc.

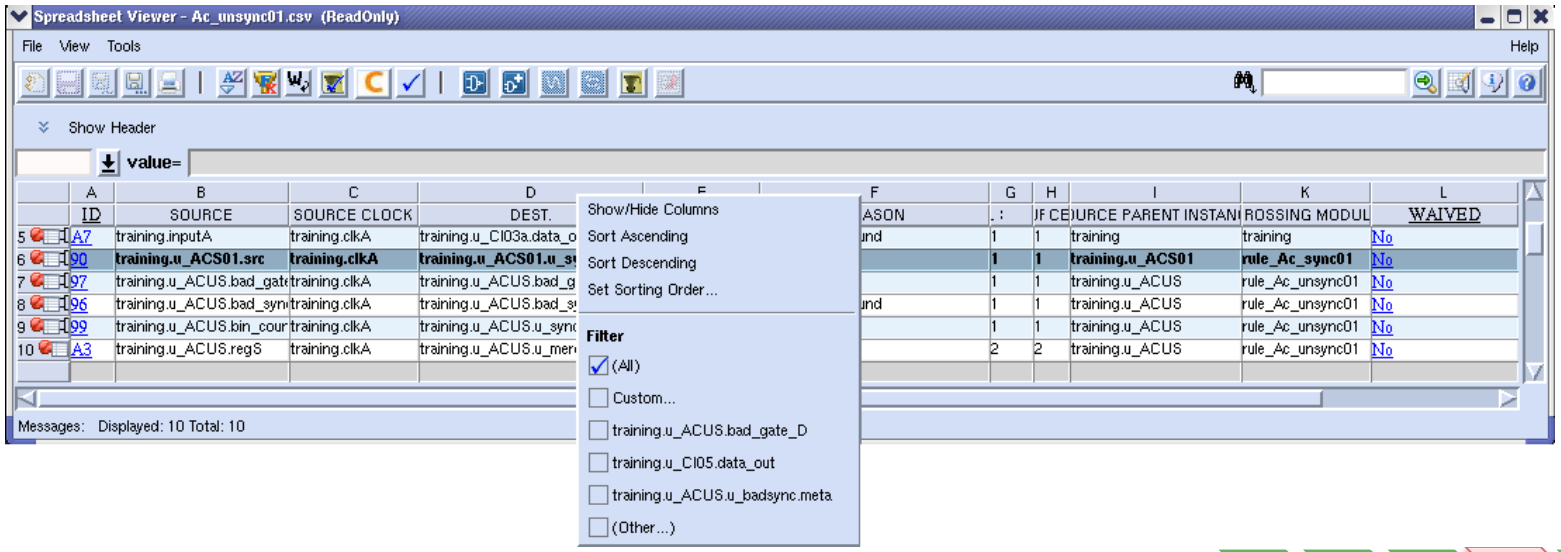
Using Spreadsheets to View CDC

SPYGLASS® FROM ATRENTA

Picture of rule-based spreadsheet (primary spreadsheet)

- Tricks and tips:
  - Sorting (reason, clocks, destinations)
  - Filtering (from/to clocks, reasons, destinations)
- Many times control synchronizers contain special names
  - Search for “meta” or “sync”
  - (And) Rule Ac_ unsync01 or 02
  - To quickly identify unsync’d control crossings



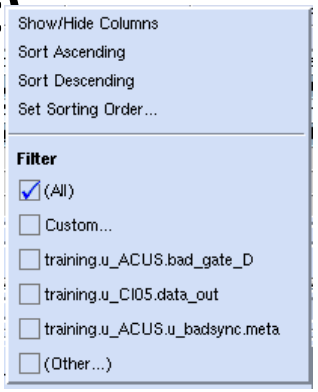


Atrenta Confidential © 2014 Atrenta Inc.

Using Spreadsheets to View CDC

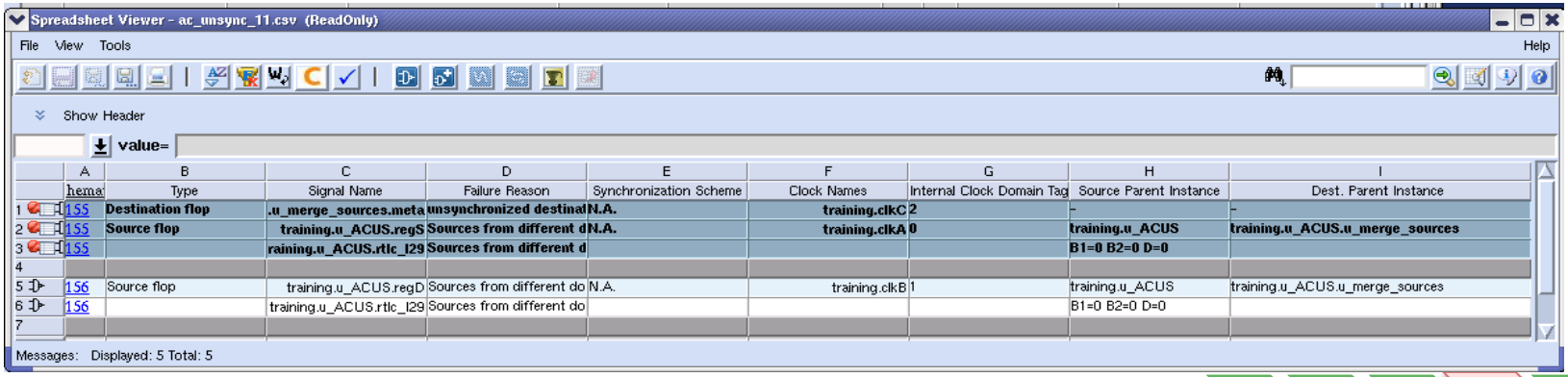
SPYGLASS® FROM ATRENTA

- **Push into violation spreadsheet (2nd spreadsheet)**



- **Columns contain:**
  - Type of Info = destination, source, input, qualifier, potential qualifier
  - Name of object – Full instance path to output NET of object
  - FAILURE REASON – ALWAYS LOOK HERE!
  - SYNC REASON – Not all sources are unsynchronized
  - Clock name(s), internal clock domain tag

- **Tricks: You can sort/search on columns just like previous**

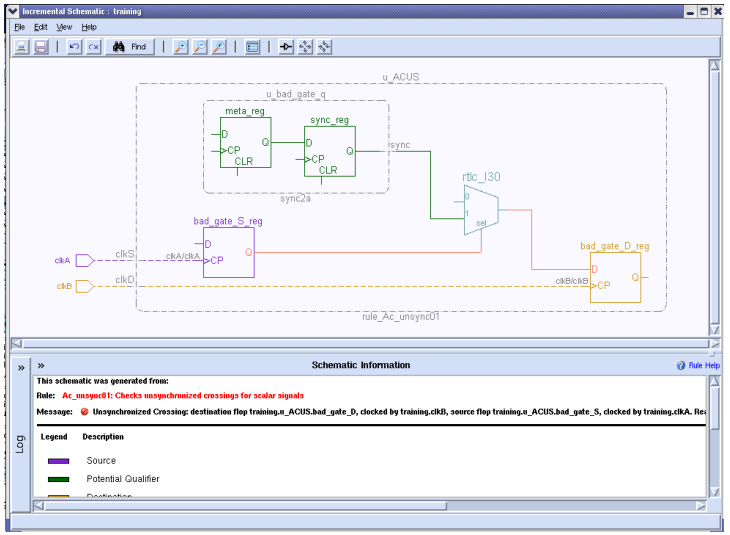


	A	B	C	D	E	F	G	H	I
	hema	Type	Signal Name	Failure Reason	Synchronization Scheme	Clock Names	Internal Clock Domain Tag	Source Parent Instance	Dest. Parent Instance
1	155	Destination flop	.u_merge_sources.meta	unsynchronized destinat	N.A.	training.clkC	2	-	-
2	155	Source flop	training.u_ACUS.regS	Sources from different d	N.A.	training.clkA	0	training.u_ACUS	training.u_ACUS.u_merge_sources
3	155		raining.u_ACUS.rtlc_I29	Sources from different d				B1=0 B2=0 D=0	
4									
5	156	Source flop	training.u_ACUS.regD	Sources from different do	N.A.	training.clkB	1	training.u_ACUS	training.u_ACUS.u_merge_sources
6	156		training.u_ACUS.rtlc_I29	Sources from different do				B1=0 B2=0 D=0	
7									

Atrenta Confidential © 2014 Atrenta Inc.

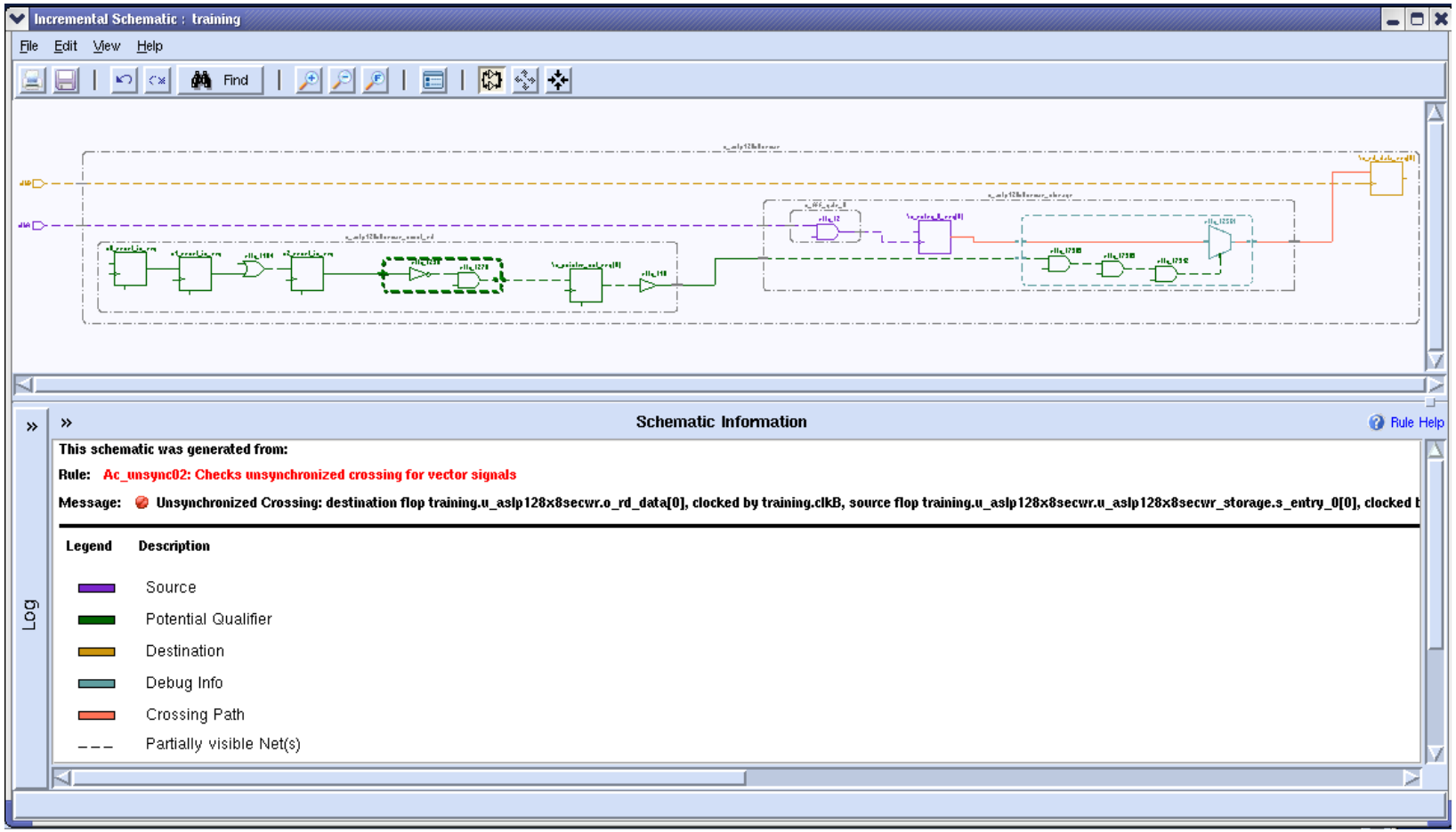
Schematics and CDC's (DEMO)

- **Colors are used to identify the domains and qualifiers – Legend is provided at bottom**
- **Info case analysis (^A) always turn on!**
- **Right click on any NET (NO PINS) and select DEBUG_DATA to see domain info**
- **Hierarchy boundaries can be expanded by (double-click) on edges**
- **Tracing inputs/outputs using double-click**
- **Tracing inputs/outputs using menus to trace to flops/latches/inputs/outputs/module**
- **Cross probing happens automatically between schematic and source code**
  - Schematic to RTL
  - RTL to schematic
- **Finding anything using find button**
- **Reports that help with CDC**
  - CDC-report
  - Ac_sync_group
  - CKSGDCInfo
  - Clock-Reset-Detail
  - Ac_sync_qualifier



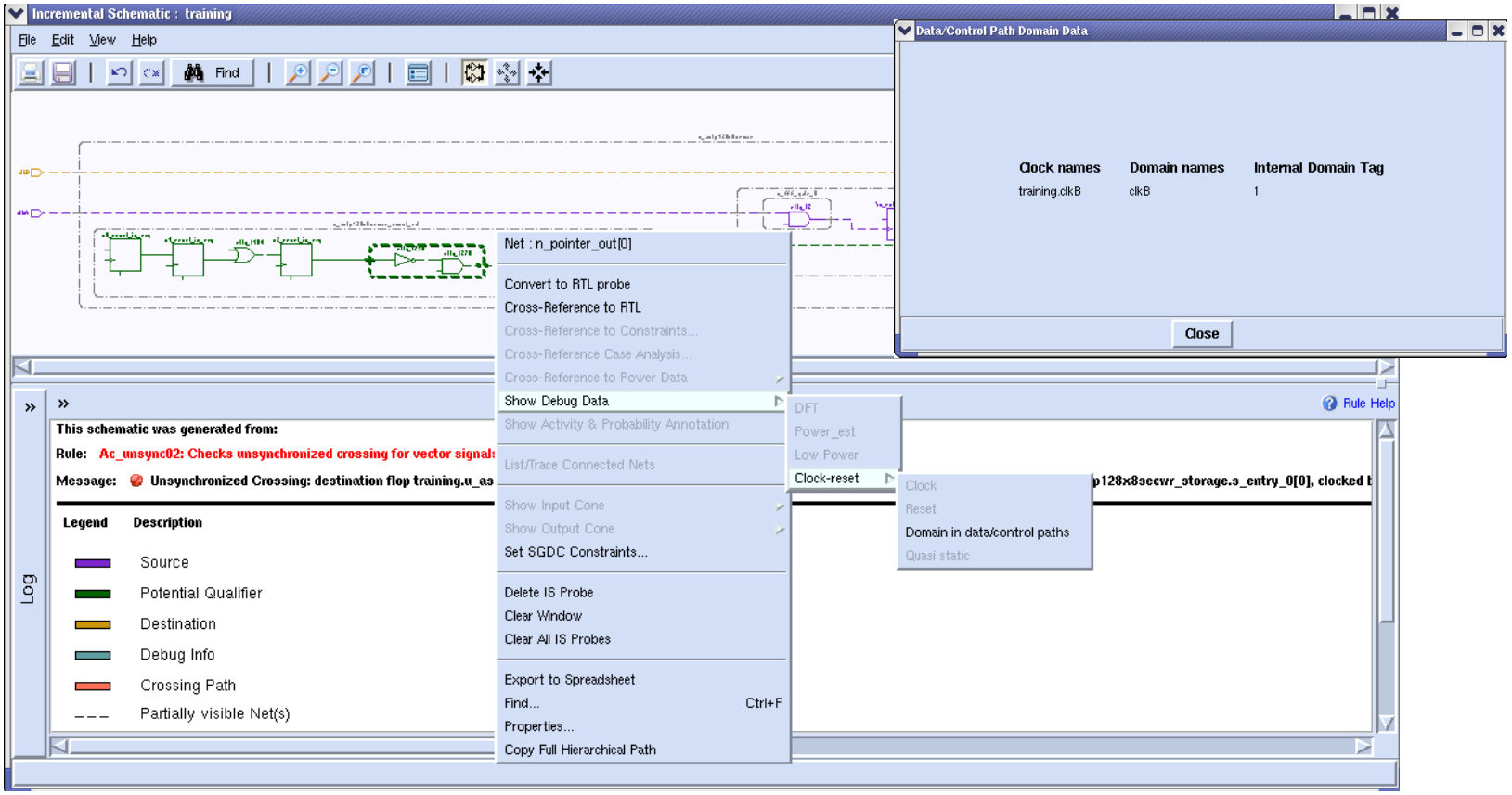
Atrenta Confidential © 2014 Atrenta Inc.

Ac_unsync Schematic Colors / Legend



Atrenta Confidential © 2014 Atrenta Inc.

Debug_data – Right Click on NET

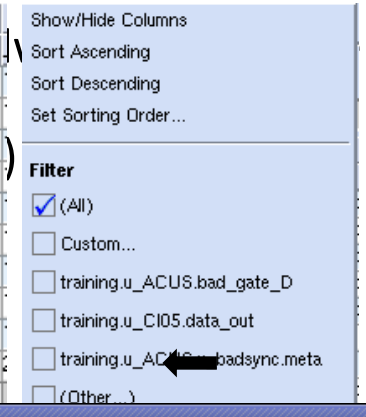


Atrenta Confidential © 2014 Atrenta Inc.

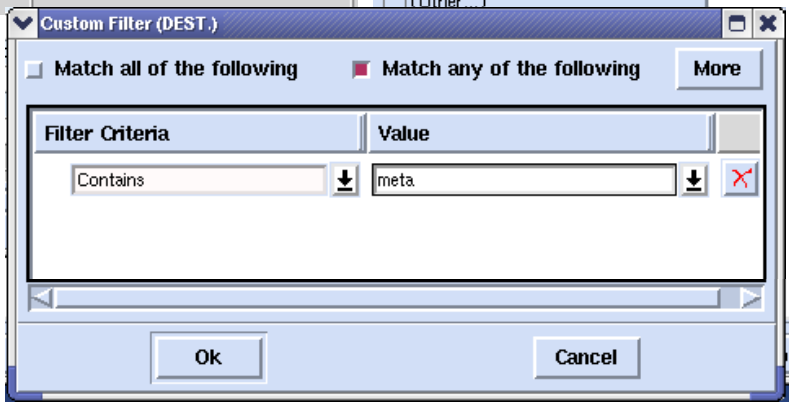
Example Debug Session (1 of 6)

- **Start with Ac_unsync01 spreadsheet – There are MANY violations**
- **The two most interesting columns to start with are the DESTINATION and REASON**

B	C	D	E	F
SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON
training.inputA	training.clkA	training.u_CI05.data_out	training.clkB	Qualifier not found
training.inputA	training.clkA	training.u_CI05.data_out1	training.clkB	Qualifier not found
training.inputA	training.clkA	training.u_ACUS.regD	training.clkB	Qualifier not found
training.inputA	training.clkA	training.u_ACUS.u_badsync.meta	training.clkB	Combinational logic used between c
training.inputA	training.clkA	training.u_CI03a.data_out	training.Clock_inf	Qualifier not found
training.u_ACS01.src	training.clkA	training.u_ACS01.u_sync2s.meta	training.clkB	Sync reset used in multi-flop synchi
training.u_ACUS.bad_gate	training.clkA	training.u_ACUS.bad_gate_D	training.clkB	Gating logic not accepted: source c
training.u_ACUS.bad_syn	training.clkA	training.u_ACUS.bad_sync_D	training.clkB	Qualifier not found
training.u_ACUS.bin_cour	training.clkA	training.u_ACUS.u_sync2s.meta	training.clkB	Sync reset used in multi-flop synchi
training.u_ACUS.regS	training.clkA	training.u_ACUS.u_merge_sources.met	training.clkC	Sources from different domains co



- Change Filter Criteria to “Contains”
- Type in Value=meta
- Click OK



Atrenta Confidential © 2014 Atrenta Inc.

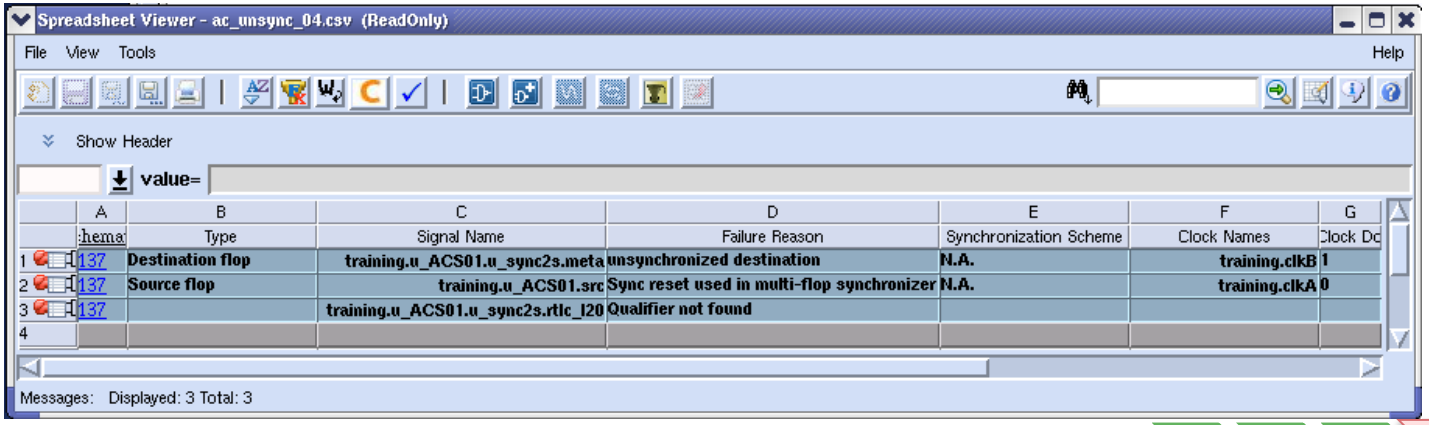
Example Debug Session (2 of 6)

SPYGLASS® FROM ATRENTA

- **After filtering only four violations are shown**
  - NOTE! Column D has little “Filter” symbol now
- **Select the violation instance:**
  - training.u_ACSO1……
  - New spreadsheet appears

B	C	D	E	F
SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON
training.u_ACS01.src	training.clkA	training.u_ACS01.u_sync2s.meta	training.clkB	Sync reset used in multi-flop synchi
training.u_ACUS.bin_cour	training.clkA	training.u_ACUS.u_sync2s.meta	training.clkB	Sync reset used in multi-flop synchi
training.inputA	training.clkA	training.u_ACUS.u_badsync.meta	training.clkB	Combinational logic used between c
training.u_ACUS.regS	training.clkA	training.u_ACUS.u_merge_sources.met	training.clkC	Sources from different domains co

- **Note the REASON:**
  - Sync reset used in multi-flop synchronizer
- **In a CONTROL CROSSING, combinatorial logic is NOT allowed since it could GLITCH**
  - Glitches are OK in a synchronous crossing since STA insures stability before clock
- **Open the incremental schematic by typing “I” or clicking the icon with + on it**

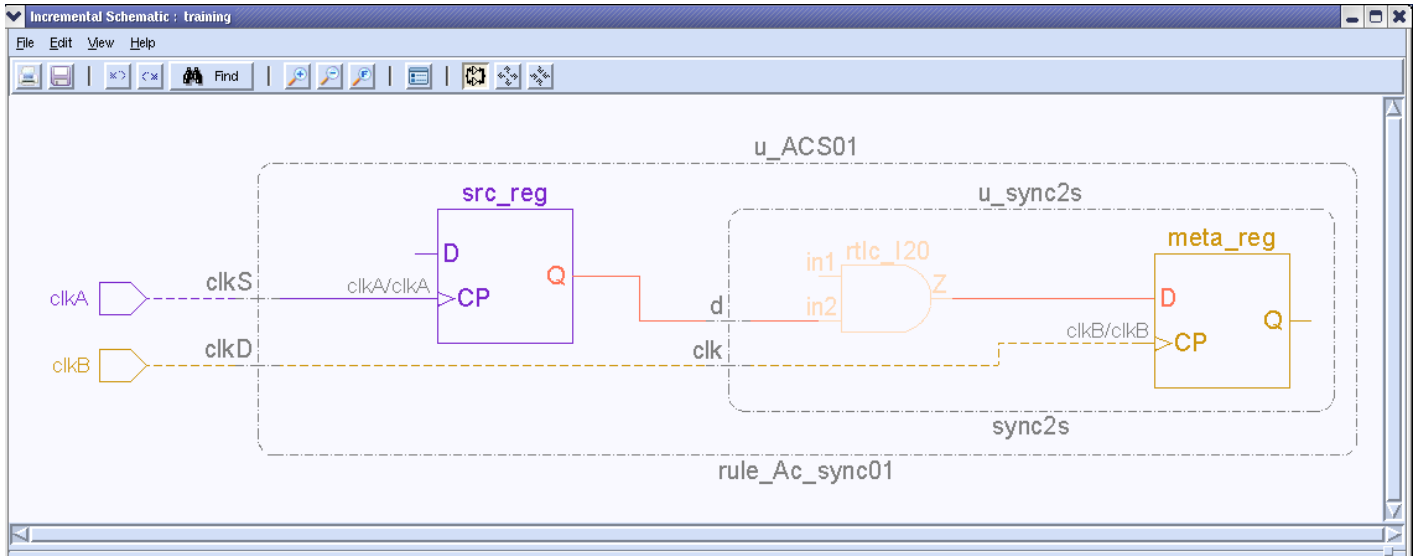


	A	B	C	D	E	F	G
	hema	Type	Signal Name	Failure Reason	Synchronization Scheme	Clock Names	Clock Dc
1	137	Destination flop	training.u_ACS01.u_sync2s.meta	unsynchronized destination	N.A.	training.clkB	1
2	137	Source flop	training.u_ACS01.src	Sync reset used in multi-flop synchronizer	N.A.	training.clkA	0
3	137		training.u_ACS01.u_sync2s.rtlc_I20	Qualifier not found			
4							

Atrenta Confidential © 2014 Atrenta Inc.

Example Debug Session (3 of 6)

- **Incremental schematic shows an AND gate in middle of crossing**
  - SpyGlass says this is wrong and guesses that the INTENT was a synchronous reset
  - NOTE! This is not always the case and must be verified



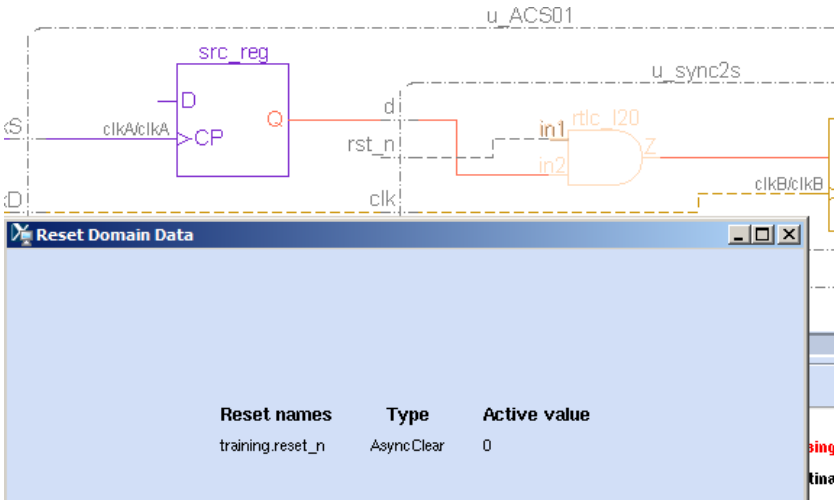
- **Double-Click on the “in1” of the AND gate and start walking it back**

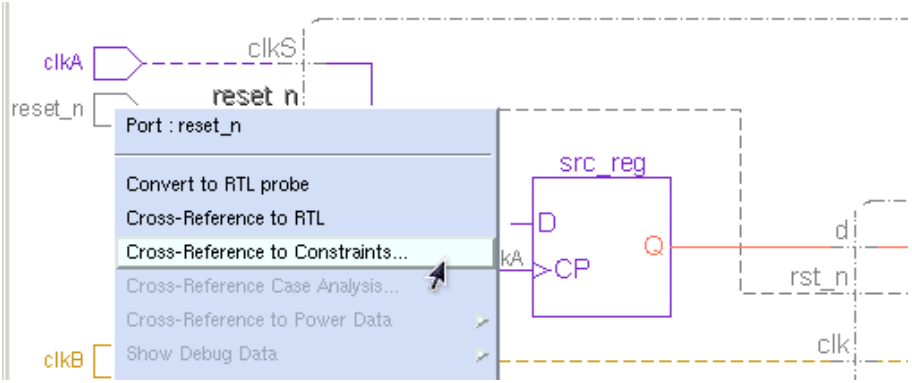
- **We can immediately see that the NET goes through the RST_N pin of the synchronizer (next page)**

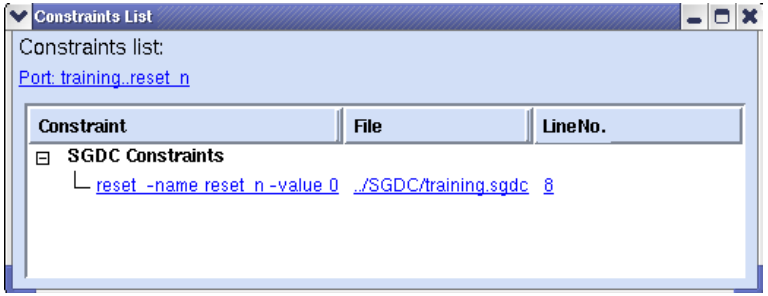
Atrenta Confidential © 2014 Atrenta Inc.

Example Debug Session (4 of 6)

- **RIGHT-CLICK on the RESET NET (not the pin) and select the DEBUG_DATA->RESET information**
  - Net is already defined
  - Net is an AsyncClear
- **You can't have an ASYNC RESET control a data reset pin, since It will cause meta-stability**
- **Follow rst_n net back to primary input**
- **RIGHT-CLICK on primary reset**
  - Cross reference to constraints
  - It shows constraint and file location
- **You can EDIT this file or create a NEW constraint on-the-fly!**





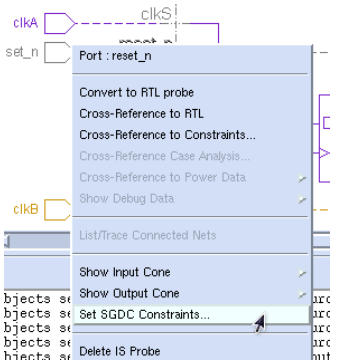


Atrenta Confidential © 2014 Atrenta Inc.

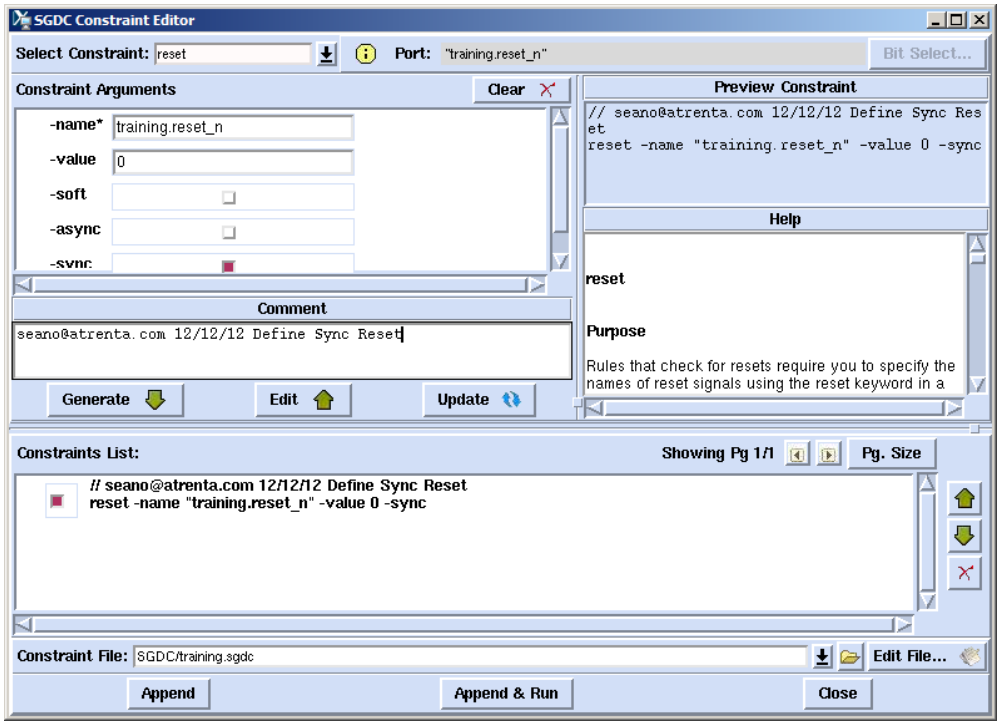
Example Debug Session (5 of 6)

SPYGLASS® FROM ATRENTA

- **Add a constraint on-the-fly**
  - RIGHT-CLICK on reset_n primary input pin
  - Click on “Set SGDC Constraints”



- **Fill out form as follows:**
  - Constraint: Reset
  - Name: AUTO FILLED
  - -SYNC: CHECK!
  - Comment:
    - Email (unique name)
    - Date
    - Reason
  - Click Generate
  - Confirm constraint file



- **Click Append & Run**

- **Violation is gone**
  - It moved to Ac_sync

Atrenta Confidential © 2014 Atrenta Inc.

Example Debug Session (6 of 6)

- **After fixing the reset –sync issue:**
  - Re-open the Ac_sync01 spreadsheet ,and,
  - Filter on meta
  - There are ONLY THREE violations now

- **Pick the REASON: Sources from different domains converge before being synchronized**
  - NOTE! A multi-flop synchronizer can only sync data from ONE domain
  - Open the incremental schematic
  - RIGHT-CLICK on NET after the AND gate and show the DEBUG_DATA->Domain
    - NOTE! It is two domains
  - Expand the source of the AND gate and see the 2nd register
    - Designer TOLD ME that regS is a “STATIC” config register that won’t change
  - Use the “Constraint Editor” gui to apply a Quasi_static constraint to the regS NET
  - APPLY and RERUN
  - Confirm that the Violation is GONE
    - It has moved to the Ac_sync folder

- ***You now have the idea of what it takes to debug CDC crossings in SpyGlass***

Atrenta Confidential © 2014 Atrenta Inc.

Philosophy of Resolving Ac_unsync Violations

SPYGLASS® FROM ATRENTA

- **Do not waive Ac_unsync01/02 violations**
  - It only waives the crossings listed in the message tree
  - It HIDES other potential crossing violations from other sources
  - If you need to reuse constraints at next higher level can sometimes be difficult.

- **Use constraints to resolve synchronization issues**
  - **set_case_analysis** – Use to insure only one clock or correct mode of operation
  - **quasi_static** – Signals that won't toggle during functional operation
    - This will resolve combo-logic issues
  - **reset** - Use in a multi-flop synchronizer
  - **qualifier** – Use when real multi-flop doesn't exist
    - Primary inputs (abstract_port –sync)
    - Blackboxes (qualifier)
  - **sync_cell** – Only necessary if SpyGlass doesn't automatically recognize it
  - **FIFO constraint**
  - **cdc_false_path** – If sync scheme is externally or software controlled
    - Examples: clocks are off during changing
    - There are other rules that will honor this constraint and remove other potential violations

SpyGlass can Recognize 99.99% of Real Crossings, So If SpyGlass Says Its Bad, It May Really Be BAD!

Atrenta Confidential © 2014 Atrenta Inc.

Handling Large Number of Violations

SPYGLASS® FROM ATRENTA

- **If the number of violations is very high (say thousands of violations), it could be due to incorrect/ incomplete setup**
  - Check setup
    - Multi clock Black-Boxes are very noisy. Fix First
    - Clock_info05/a – Missing case_analysis on Muxes is very noisy
- **Open the spreadsheet of this rule**
  - RIGHT-CLICK on the rule header in the message tree, and select the Open Spreadsheet option from the shortcut menu
  - Spreadsheet provides an easy way to navigate through large number of violations
  - Cross-probe to RTL code and view a schematic of the violation
  - Use filtering and sorting to debug the related messages
  - Waive or apply cdc_false_path constraint directly from the spreadsheet

Atrenta Confidential © 2014 Atrenta Inc.

Block CDC Verification - Ac_unsyncXX

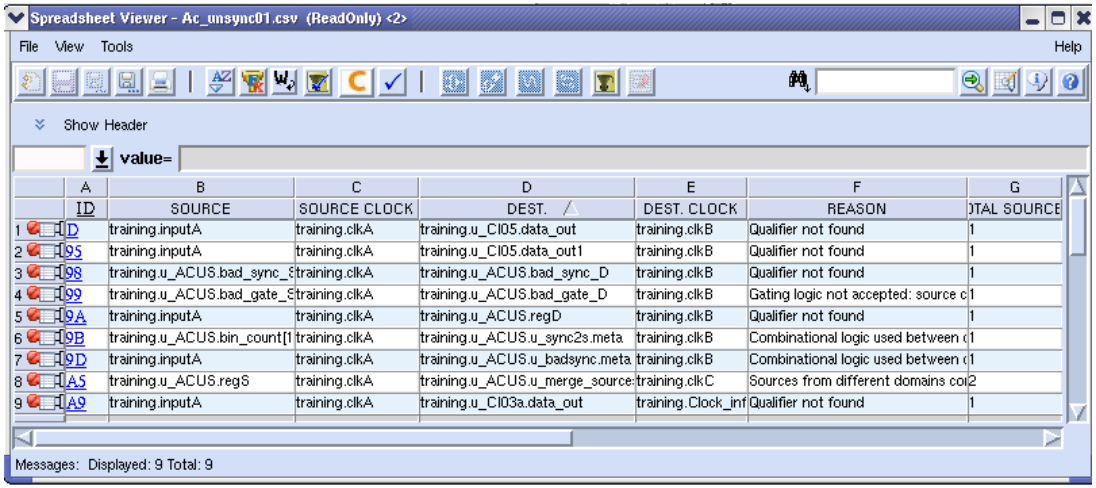
SPYGLASS® FROM ATRENTA

Debug Ac_unsyncXX (control crossings) first

- Ac_unsync01 flags CDC on SCALAR signals.
- Ac_unsync02 flags CDC on VECTOR signals.
  - A single bit Vector is still a Vector (Example: meta[1] is a vector).
- Usually, but not always, Scalar Crossings are Control Crossings.
- Know your design style to decide to look for control crossings in Ac_unsync02.

Open rule spreadsheet to see all destinations of Ac_unsync01

- Each row in the spreadsheet shows unique destination of crossing
- It will show one of the source for a destination, refer violation spreadsheet to see all sources

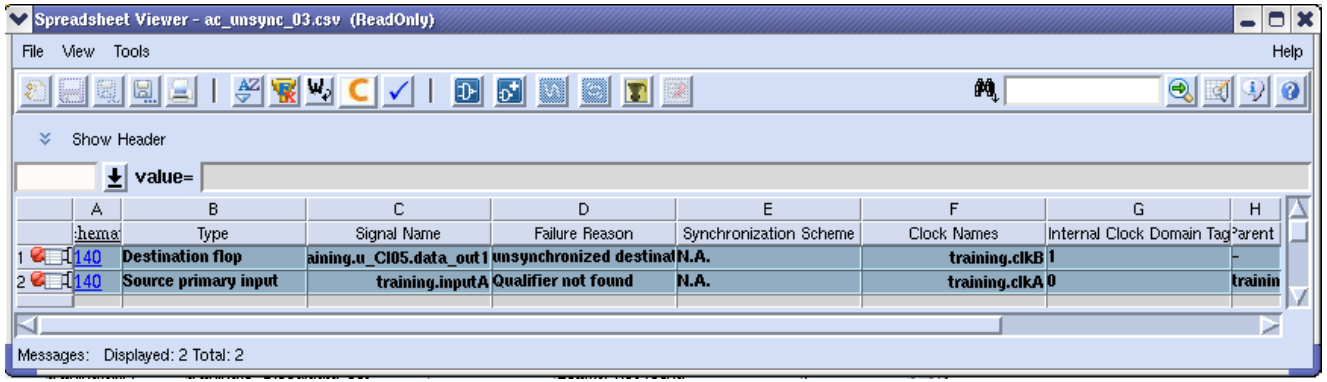


	A	B	C	D	E	F	G
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON	OTAL SOURCE
1	D	training.inputA	training.clkA	training.u_CI05.data_out	training.clkB	Qualifier not found	1
2	95	training.inputA	training.clkA	training.u_CI05.data_out1	training.clkB	Qualifier not found	1
3	98	training.u_ACUS.bad_sync_S	training.clkA	training.u_ACUS.bad_sync_D	training.clkB	Qualifier not found	1
4	99	training.u_ACUS.bad_gate_S	training.clkA	training.u_ACUS.bad_gate_D	training.clkB	Gating logic not accepted: source c	1
5	9A	training.inputA	training.clkA	training.u_ACUS.regD	training.clkB	Qualifier not found	1
6	9B	training.u_ACUS.bin_count[1	training.clkA	training.u_ACUS.u_sync2s.meta	training.clkB	Combinational logic used between c	1
7	9D	training.inputA	training.clkA	training.u_ACUS.u_badsync.meta	training.clkB	Combinational logic used between c	1
8	A5	training.u_ACUS.regS	training.clkA	training.u_ACUS.u_merge_source	training.clkC	Sources from different domains co	2
9	A9	training.inputA	training.clkA	training.u_CI03a.data_out	training.Clock_inf	Qualifier not found	1

Atrenta Confidential © 2014 Atrenta Inc.

Block CDC Verification - Ac_unsyncXX (cont'd)

- **Open violation spreadsheet by clicking on ID field**



- **Violation spreadsheet will represent the "Complete Architecture" of the crossing**
  - Showing all sources for that destination
  - Listing which sources are unsynchronized with failure reason in column D
  - Listing which sources are synchronized with synchronization scheme in column E
  - Qualifiers detected for synchronized crossings
  - Potential qualifiers as debug hint for unsynchronized crossing (if they exist)

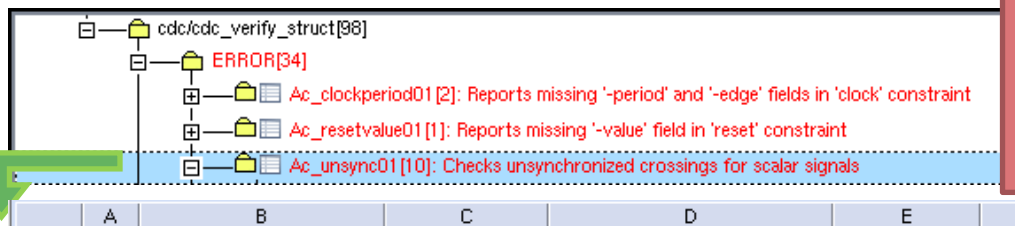
- **Fix Ac_unsyncXX violations**
  - In case it is missing synchronizer, add the synchronizer
  - Custom sync cell
  - Exception

Atrenta Confidential © 2014 Atrenta Inc.

Debugging Ac_unsyncXX

SPYGLASS® FROM ATRENTA

- View offending crossing, reason, stats in Summary spreadsheet



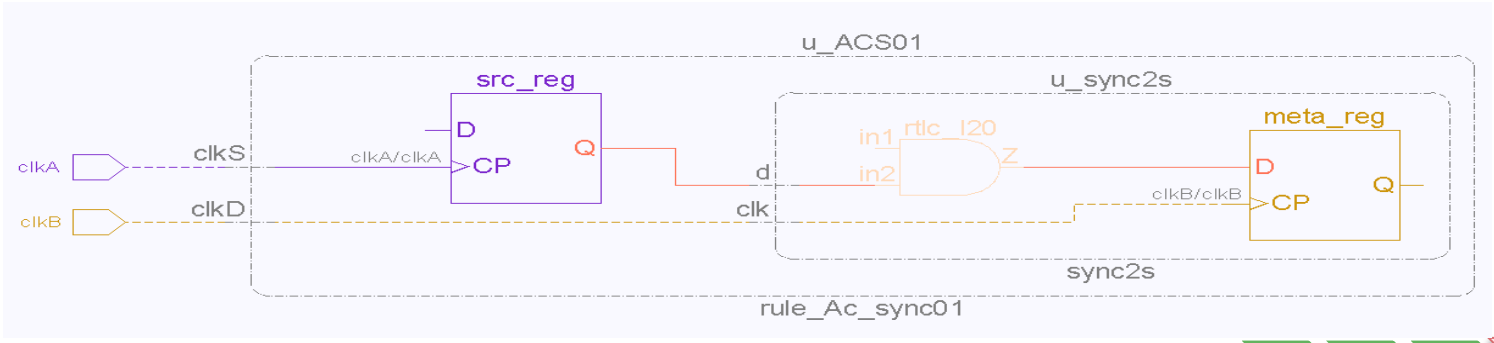
IMPORTANT NOTE
Ac_sync rules in message tree & summary spreadsheet only report one source. To see all sources review the detailed spreadsheet

	A	B	C	D	E	F	G	H
	ID	SOURCE	SOURCE CLOCK	DEST.	DEST. CLOCK	REASON	TOTAL SOURCES	OTAL SOURCE DOMAIN
8	A3	training.u_ACUS.regS	training.clkA	training.u_ACUS.u_merge_source:	training.clkC	Sources from different domains co	2	2
9	90	**training.u_ACS01.src**	**training.clkA**	**training.u_ACS01.u_sync2s.met**	**training.clkB**	**Sync reset used in multi-flop sy**	**1**	**1**

- Link to crossing architecture in Detail spreadsheet

	A	B	C	D	E	F	G
	hema	Type	Signal Name	Failure Reason	Synchronization Scheme	Clock Names	ck
1	137	**Destination flop**	**training.u_ACS01.u_sync2s.meta**	**unsynchronized destination**	**N.A.**	**training.clkB**	**1**
2	137	**Source flop**	**training.u_ACS01.src**	**Sync reset used in multi-flop synchronizer**	**N.A.**	**training.clkA**	**0**
3	137		**raining.u_ACS01.u_sync2s.rtlc_I20**	**Qualifier not found**			

- Cross-highlight individual crossing paths in Incremental Schematic

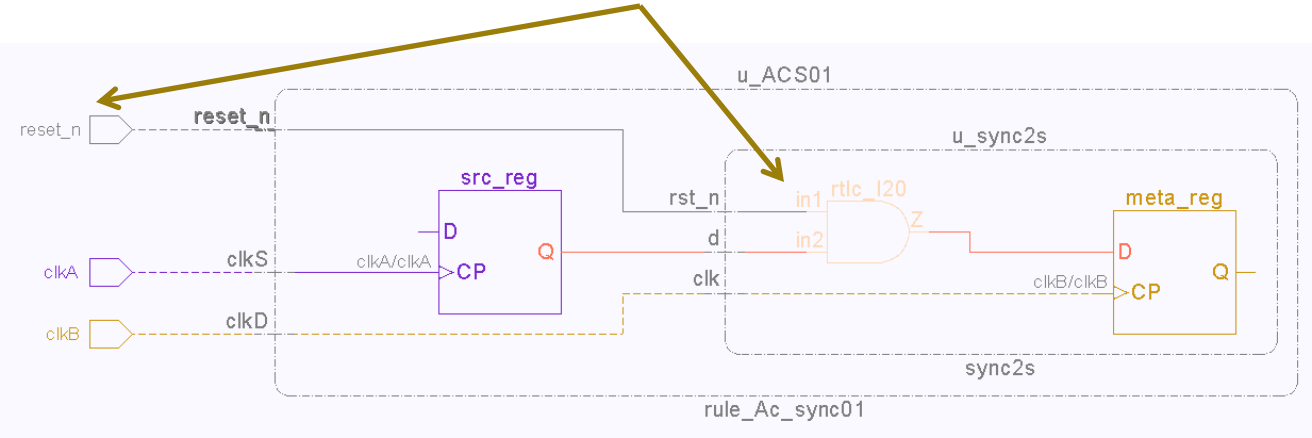


Atrenta Confidential © 2014 Atrenta Inc.

Specify Missing Synchronous Resets

SPYGLASS® FROM ATRENTA

Explore fanin to identify sync reset net



Specify SGDC constraint reset on sync reset net

```
38
39 ##Synchronous Reset Signals
40
41 reset -name reset_n -sync -value 0
```

IMPORTANT NOTE
Declaring as sync reset implies reset is static and will not jeopardize intended metastability synchronization; implies that when reset changes, metastability synchronization is not a concern

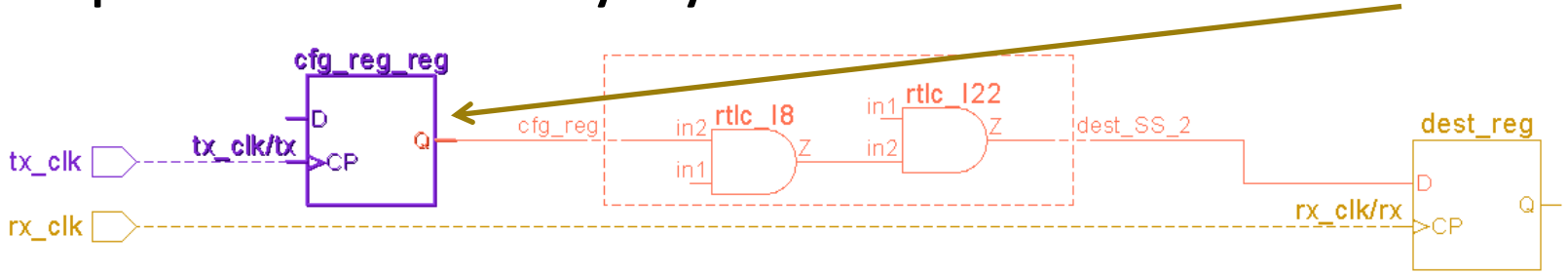
Confirm desired effect (on to next Ac_unsync01 failure reason)

	A	B	C	D	E	F
	hema	Type	Signal Name	Failure Reason	Synchronization Scheme	Clock Names
1	148	Destination flop	g.u_ACUS.u_badsync.meta	unsynchronized destination	N.A.	training.clkB
2	148	Source primary input	training.inputA	Combinational logic used between crossing	N.A.	training.clkA
3	148		training.u_ACUS.rtlc_l28	Qualifier not found		

Atrenta Confidential © 2014 Atrenta Inc.

Specify Missing Quasi-Static Constraints

- When failure reason is “Combinational logic used between crossing,” explore sources to identify any that should be considered static



- Specify SGDC constraint `quasi_static` on static net(s)

IMPORTANT NOTE
Declaring as quasi-static implies signal is static and will not jeopardize intended metastability synchronization; implies that when signal changes, metastability synchronization is not a concern (typically the destination clock is stopped or the path is blocked)

```
##=============================================
## QUASI-STATIC SIGNALS
##---------------------------------------------
// Avoid Ac_unsync01 "Combo logic in crossing"
quasi_static -name "cfg_reg[0]"
##=============================================
```

- Confirm desired effect (in Ac_sync01 INFO message)

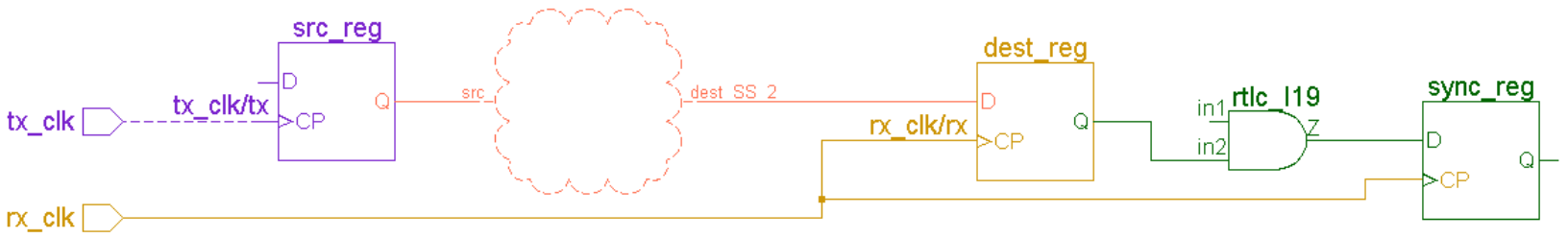
	A	B	C	D	E
	Schematic	Type	Signal Name	Synchronization Scheme	Clock Names
1	1	Destination flop	design1.dest	Conventional multi-flop for metastability technique	design1.rx_clk
2	1	Source flop	design1.src	Conventional multi-flop for metastability technique	design1.tx_clk
3	2	Source flop	design1.cfg_reg	does not require synchronization (long-delay/quasi-static)	design1.tx_clk

Atrenta Confidential © 2014 Atrenta Inc.

Crossing Synchronized for Metastability

- See recognized synchronization flop (green) in incremental schematic

	A	B	C	D	E
	Schematic	Type	Signal Name	Synchronization Scheme	Clock Names
1	1	Destination flop	design1.dest	Conventional multi-flop for metastability technique	design1.rx_clk
2	1	Source flop	design1.src	Conventional multi-flop for metastability technique	design1.tx_clk
3	2	Source flop	design1.cfg_reg	does not require synchronization (long-delay/quasi-static)	design1.tx_clk



Atrenta Confidential © 2014 Atrenta Inc.

Why Waivers Are Not Recommended for Ac_unsync

- **Glitch, Convergence and Functional Checks are NOT done on Unsynchronized crossings.**
- **Data Crossings require a GOOD control crossing.**
  - If Control Crossing is reported UN-synchronized,
  - then related DATA crossing will be reported Un-synchronized
- **Use proper design constraints (quasi_static, reset, set_case_analysis, input, signal_in_domain, etc.) in order for Synchronzied Crossings to get reported as synchronized.**
- **Unsynchronized control crossings will not be checked for Gitches (Ac_glitch03), Fast2Slow (Ac_cdc01a), DataHold (Ac_datahold01a), and Coherency Checks (Ac_convXX)**
- **If the crossing was never meant to be synchronized, then it is OK to waive.**
- **All other types of violations except Ac_unsyncXX are usually OK to waive.**
  - Except violations that tell you that you are missing Clocks/Resets

Atrenta Confidential © 2014 Atrenta Inc.

Additional Recommendations (CDC Exceptions)

SPYGLASS® FROM ATRENTA

There are two types of unsynchronized crossings:

1) Intentionally left unsynchronized
   - Configuration registers
   - Software controlled crossings
   - Use quasi_static constraint to remove from analysis
     - » It is self documenting
   - DON'T USE WAIVERS
     - » Only first source is reported/waived
     - » You could miss the REAL crossing

2) Intentionally synchronized but assumption(setup) missing
   - Use constraints to provide assumption to the tool
     - » One source is left unsynchronized, but others are sync
     - » Like GPIO (interrupts)
   - Static signals – declare quasi_static
   - Undeclared synchronous reset
   - Don't waive or cdc_false_path a synchronized crossing
     - » You will miss the following checks:
     - » Data coherency, data hold, data crossings

Atrenta Confidential © 2014 Atrenta Inc.

Block CDC Verification – Ac_unsync02

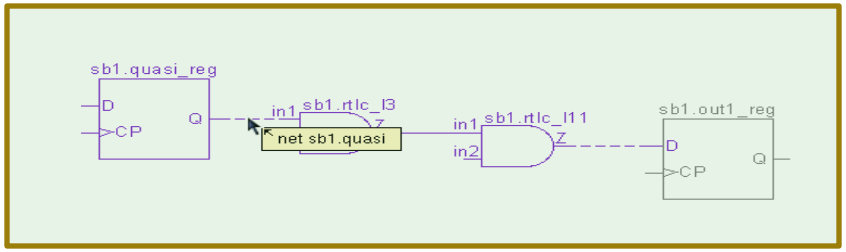
SPYGLASS® FROM ATRENTA

- **Resolve Control Crossings and re-running the block verification**
  - Ensure there are 0 violations of Control Crossings before debugging Data Crossings.
  - Ac_unsync01 flags clock domain crossings for scalar signals
- **Analyze and fix Ac_unsync02 (typically data-bus) violations**
  - Refer to the earlier slide of the architectural view of the crossing which shows how resolving the qualifier will automatically synchronize data and address
  - Use debug hint from failure reason and potential qualifier
- **Run and follow existing CDC methodology guide for cdc_verif template**

Atrenta Confidential © 2014 Atrenta Inc.

Static Signals: Debug Process

- **Static flops are the leading cause of false synchronization violations**
  - Debugging such violations is not easy as there is no sign of synchronization, and SpyGlass CDC cannot give any hints
  - Often you don't have a list of quasi-static flops – you often look at violations and then waive them as quasi-static



- **How to investigate it, and who to ask for more information**
  - Ideally, static register definitions should be covered during setup
  - Add rule `Setup_quasi_static01` to project file to find signals that are likely to be quasi-static.
  - The rule Infers quasi-static signals in a design, and generates `quasi_static` constraints for such signals in `auto_quasi_static.sgdc` file.
  - You must review the quasi_static constraints in this file and use them in the subsequent SpyGlass CDC runs.
  - Quasi-static information is also typically available with the person or team responsible for software control of the chip
  - Add the `quasi_static` constraint for these signals as follows
    - Example syntax: `quasi_static -name <net-name>`

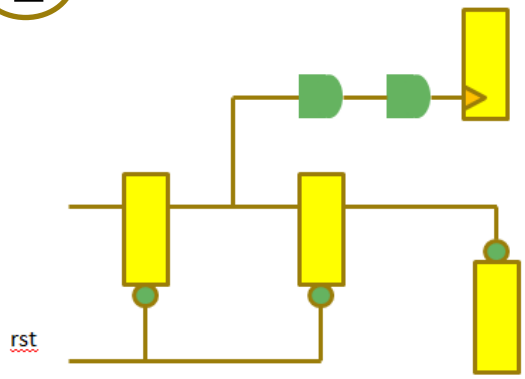
Atrenta Confidential © 2014 Atrenta Inc.

Reset Synchronization: Ar_unsync01

- Reports reset pins of flip-flops that do not have a reset synchronizer in any of the paths between an asynchronous reset source and flops
- Design scenarios reported are :

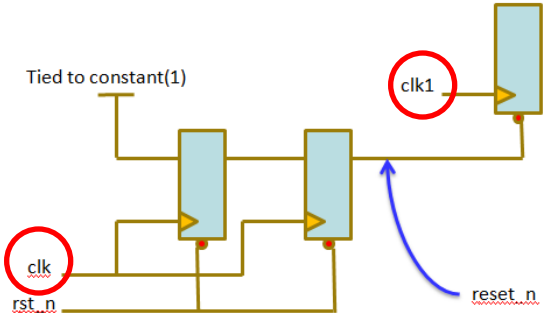
1. Missing synchronizer

2. Invalid synchronizer



Multiple fan-out of flip-flops in a synchronizer chain are present

3. Different domain synchronizer



Atrenta Confidential © 2014 Atrenta Inc.

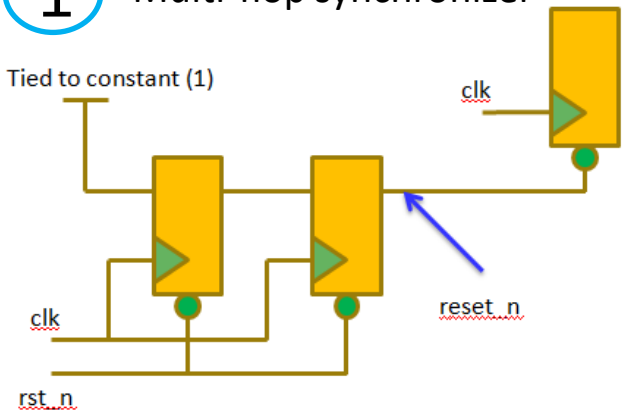
Reset Synchronization: Ar_unsync01 (Contd)

How to debug and fix ?

- To fix this violation, add a proper synchronization circuit for the reset signal

Reset synchronization schemes

1 Multi-flop synchronizer



2 Reset constrained with input constraint

```
input -name rst_n -clock clk
```

The clock domain specified by the input constraint is matched with the clock domain of the flip-flop where it is used as a reset

3 User-defined reset synchronizer

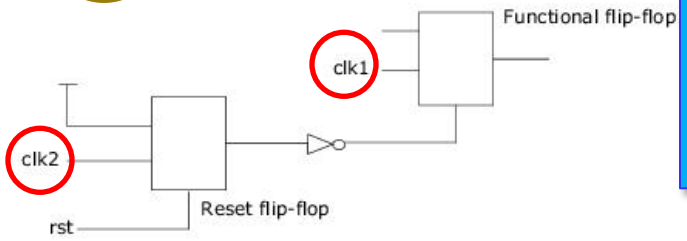
1. Incase a synchronizer scheme is not recognized by SpyGlass, use constraint `reset_synchronizer` to specify the synchronizer signal
2. Use synchronizer cells and specify them using parameter: `reset_synchronize_cells`

Atrenta Confidential © 2014 Atrenta Inc.

Reset Synchronization: Ar_asyncdeassert01

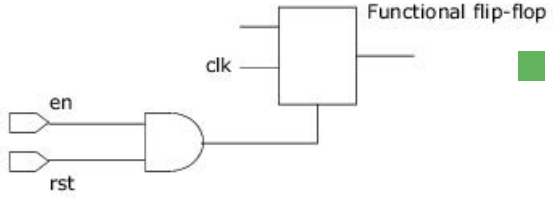
- **Reports if reset signal is asynchronously de-asserted**
- **Scenarios reported:**

1 Domain-mismatch



Domain mismatch occurs when:
1. Clock domain of reset flip-flop does not match clock domain of a functional flip-flop
2. If reset is constrained by using the `input` constraint and the clock specified by the `-clock` argument is not in the domain of the clock of the functional flip-flop

2 Improper de-assertion



Functional Flip-Flop is not synchronously de-asserted for the inactive value of the reset signal

3 Missing synchronizer

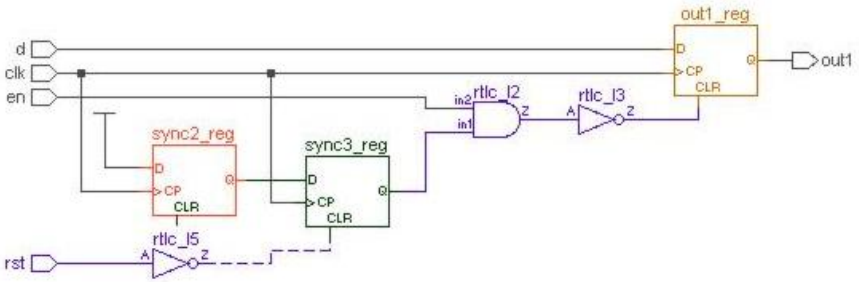
- **How to debug and fix**
  - Use the `reset_synchronizer` constraint to specify a net/port <check from definition>
  - Use the `reset_synchronize_cells` parameter to specify a black box, library cell, or a module that has a custom reset synchronizer.
  - Use the `input` constraint to specify input constraint on reset source if it is synchronized outside the block.

Atrenta Confidential © 2014 Atrenta Inc.

Reset Synchronization: Ar_sync01

- **Reports synchronized reset signals in the design**
- **Synchronization methods reported by the rule-**
  - Multi-flop reset synchronizer
  - Reset constrained by using the `input` constraint
  - User-defined reset synchronizer

Example



In the schematic, **a multi-flop reset synchronizer** is present between the reset source and flip-flop

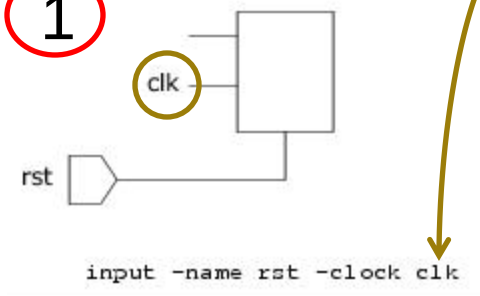
- **Schematic Highlight**
  - Following details are highlighted in different colors:
    - Reset path
    - Destination flop
    - Reset synchronizer used for synchronizing a flop
- **This is a `Info only` rule - No action is required**

Atrenta Confidential © 2014 Atrenta Inc.

Reset Synchronization: Ar_syncdeassert01

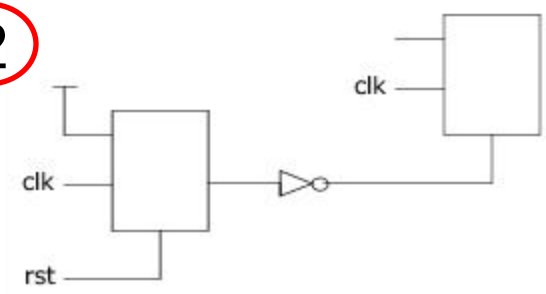
- **Reports if reset signal is synchronously de-asserted or not de-asserted at all**
- **Scenarios flagged**

1



1. Reset is specified by using the input constraint
2. And the domain of the flip-flop matches the 'clock' of the `input` constraint

2



1. At least one reset flip-flop exists in the path of a reset and a functional flip-flop
2. The functional flip-flop is de-asserted appropriately for the inactive value of the specified reset

3

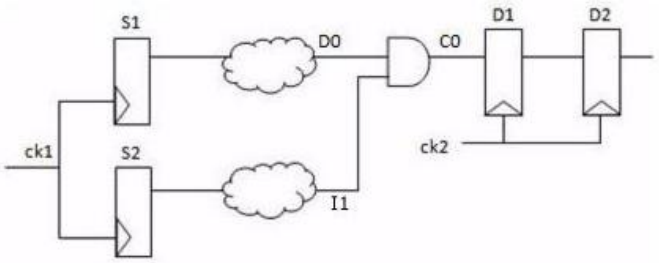
If its path is blocked such that the functional flip-flop is never de-asserted

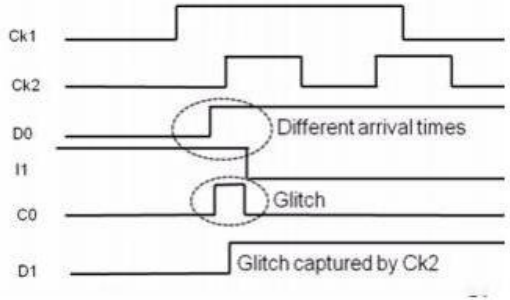
- **This is an `Info only` rule. No action is required**

Atrenta Confidential © 2014 Atrenta Inc.

Glitch Checks: rule Ac_glitch03

- **Reports clock domain crossings subject to glitches**





The glitch captured on D1 propagates to D2 and then to the downstream logic, potentially causing a functional failure

- **Checks glitch prone combinational logic in crossings synchronized by one of the following:**
  - Conventional `Multi-Flop Synchronization Scheme`
  - `Synchronizing Cell` Synchronization Scheme
  - Qualifier Synchronization Scheme Using `qualifier -crossing`

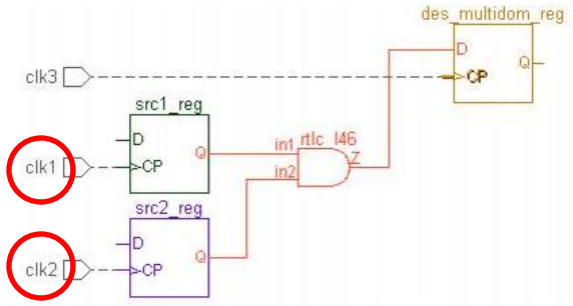
- **Rule usage guideline**
  - Set parameter `allow_combo_logic` to `yes` to perform glitch-related checks on such crossings

Atrenta Confidential © 2014 Atrenta Inc.

Glitch Checks: rule Ac_glitch03 (Contd)

Design patterns checked …

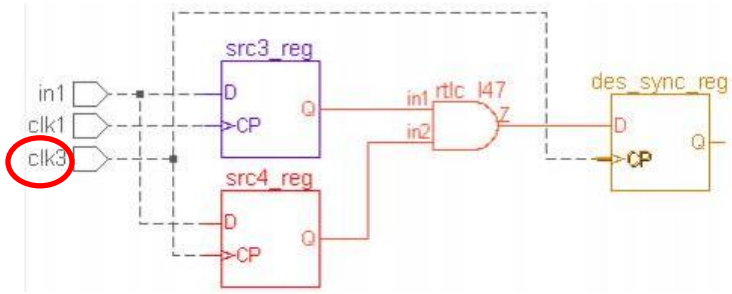
1. Sources from different domains in fanin



To fix such violations:

1. Ensure that the setup of clocks is correct
2. Check if a source in other domain is quasi-static

2. Signals from destination domain in fanin



To fix such violations:

1. Ensure that setup for clocks is correct and check if synchronous source is quasi-static
2. Else, first synchronize the asynchronous source before it converges with synchronous sources

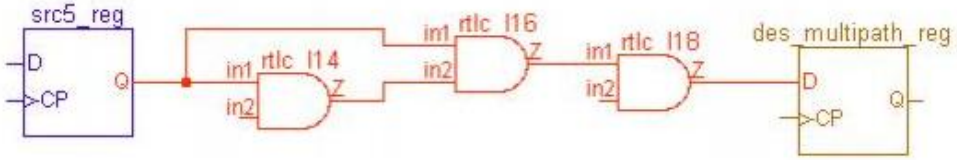
Atrenta Confidential © 2014 Atrenta Inc.

Glitch Checks: rule Ac_glitch03 (Contd)

SPYGLASS FROM ATRENTA

Design Patterns Checked

3. Source re-converges

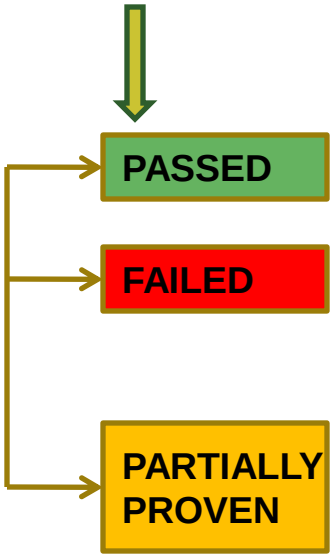


To fix such violations:

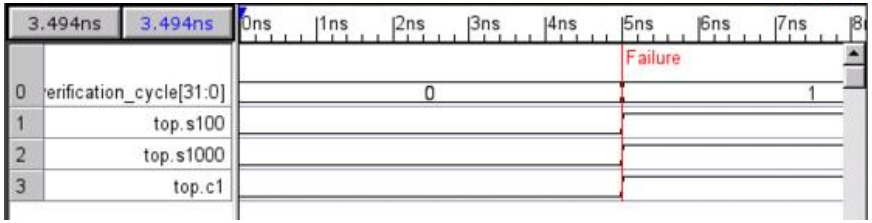
1. Modify the design so that the reported source reaches its destination through a single path
2. Ensure that timing delays of all paths from where source reaches the destination are same

4. Multiple same domain sources are toggling in a control crossing

1. Gray-encoding check is performed to ensure that at the most only one source is changing
2. This is a functional check, which is enabled only when parameter `fa_audit` is set to `no`



- No action required. Informational message
- Open waveform-viewer for the violation
- Point of failure is marked on the waveform-viewer



- At this point both asynchronous-signals are changing from 0 to 1, violating gray-encoding check
- Follow methodology for analyzing partially-proven results

Atrenta Confidential © 2014 Atrenta Inc.

Additional Recommendations (Structural)

- **Use cdc_false_path –from/-to**
  - When source should otherwise be excluded from verification
  - On a CDC that will NEVER be synchronized (static path, test)
- **Use set_parameter enable_debug_data yes to enable schematic right-click-menu access to clock, domain info for each net**
  - This is enabled in GuideWare2.0
- **Use Ac_sync_group_detail report for listing of all crossing architectures (scalar and vector destinations) and their pass/fail status**
- **Change spreadsheet column alignments to right justification**
  - Allows easy viewing of signal names

Atrenta Confidential © 2014 Atrenta Inc.

Quiz

1. **Is it safe to use double flop synchronizers to pass multi-bit data across asynchronous clock domains?**

   NO! It will create coherency problems. That is why you use DATA synchronizers with a single multi-bit qualifier.

2. **Name three synchronization scheme for passing multiple bit data across asynchronous clock domains**

   FIFO, Mux-Select, And-Scheme, Glitch-Protect.

3. **Why is de-assertion of reset asynchronously unsafe?**

   Violates Clock->Reset (Setup/Hold time) and will cause metastability.

Atrenta Confidential © 2014 Atrenta Inc.

Hands-on Exercise Lab 3

- **Hands-on exercise objectives:**
  - Lab3: CDC structural verification
    - ✓ Use SpyGlass to identify clock domain crossings that cause metastability in the design
    - ✓ Use the spreadsheet viewer in sorting CDC violations
    - ✓ Use the cdc_false_path constraint to prevent SpyGlass from reporting messages along specified paths

- **Duration:** 60 minutes

Atrenta Confidential © 2014 Atrenta Inc.

Goal: cdc/cdc_verify

Purpose

- The rules in this goal use **formal verification techniques** to discover functional issues in clock domain crossings.

Pre-requisites

- Functional checks will only run on properly synchronized logic
  - If a crossing is reported as unsynchronized, then SsyGlass will not run formal checks. For example, fast to slow or datahold checks will not be run
  - That is why we want GOOD CDC crossings reported as Ac_sync01/02
- Zero errors and zero warning in the previous goal cdc/cdc_verify_struct
  - Clocks, resets, and mode settings properly set
  - All structural issues resolved

Command

- *%shell> spyglass -project lab-CDC_proc.prj -goals cdc/cdc_verify -batch*

Output

- *List of possible clocking problems reported as rule violations*
- *Debug each violation in the rule order, iterate running SpyGlass*

Atrenta Confidential © 2014 Atrenta Inc.

cdc_verify Goal: Formal Rules

Rule	Description
Ac_datahold01a	Checks functional synchronization of synchronized data crossings
Ac_cdc01a	Checks data loss from fast-to-slow multi-flop or sync-cell synchronized clock domain crossings
Ac_conv02	Reports same-domain signals that are synchronized in the same destination domain and converge before sequential elements
Ac_conv04	Checks all the control-bus clock domain crossings that neither converge nor follow gray encoding
Ac_glitch03	Reports clock domain crossings subject to glitches
* Ac_fifo01	Reports a violation if there is an overflow or underflow in any FIFO of the design
* Ac_handshake01	Checks data loss in handshake protocol
* Ar_handshake02	Checks whether request and acknowledgement signals are following the handshake protocol or not

NOTE: This is not an exhaustive list. For a complete list of rules refer to cdc_verify.spq in GuideWare 2.0

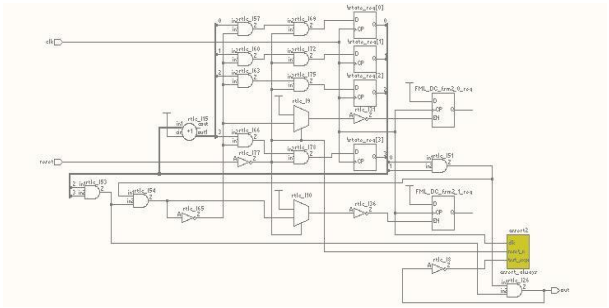
**** These are checked as part of Ac_datahold01a and are not required or recommended.***

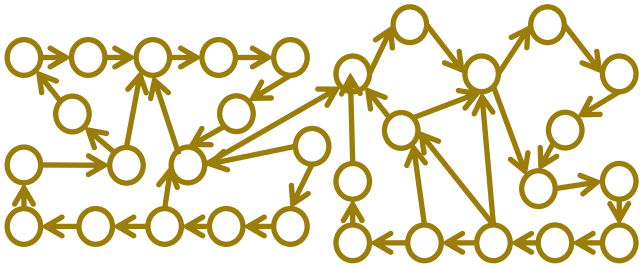
Atrenta Confidential © 2014 Atrenta Inc.

Formal Verification

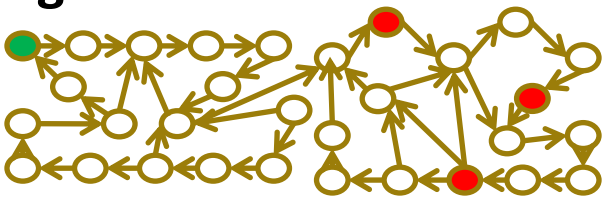
SPYGLASS® FROM ATRENTA

- **Formal verification: tackling the limitations of simulation**
- **View this circuit as a BIG finite state machine**
  - Example: If a design has 1000 flops, it is a BIG FSM with **$2^{1000} = 10^{301}$ states**





- **Visit states of the machine in search of a bug**
  - Start from an initial state – colored GREEN
  - Search for a bug state – colored RED
    - If path is found, there is a BUG, report **FAIL**
    - If no path is found, no bug found, report **PASS**
    - If time/memory resources are UP, report **partially-proven (PP)**

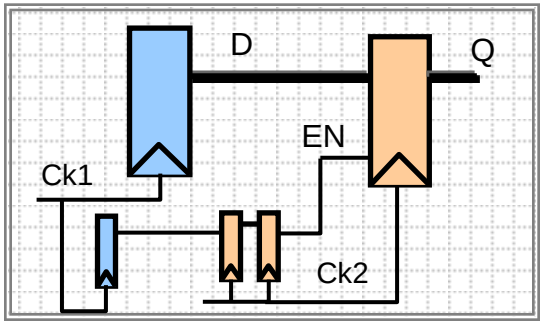


Atrenta Confidential © 2014 Atrenta Inc.

Functional CDC Verification

Why functional verification is needed?

- Well-accepted structure will not stop metastability if D changes when EN is high
- Many signals in the enable path may have multi-flop sync, not all of them are designed to prevent metastability
- Only functional verification can ensure correctness



MUST-Run functional checks – cdc_verify goal

- Data hold check for data crossings – Ac_datahold01
- Width extender for control crossings – Ac_cdc01
- Exclusivity of correlated signals – Ac_conv02

NICE to run checks

- FIFO overflow/underflow checks
- Handshake protocol check

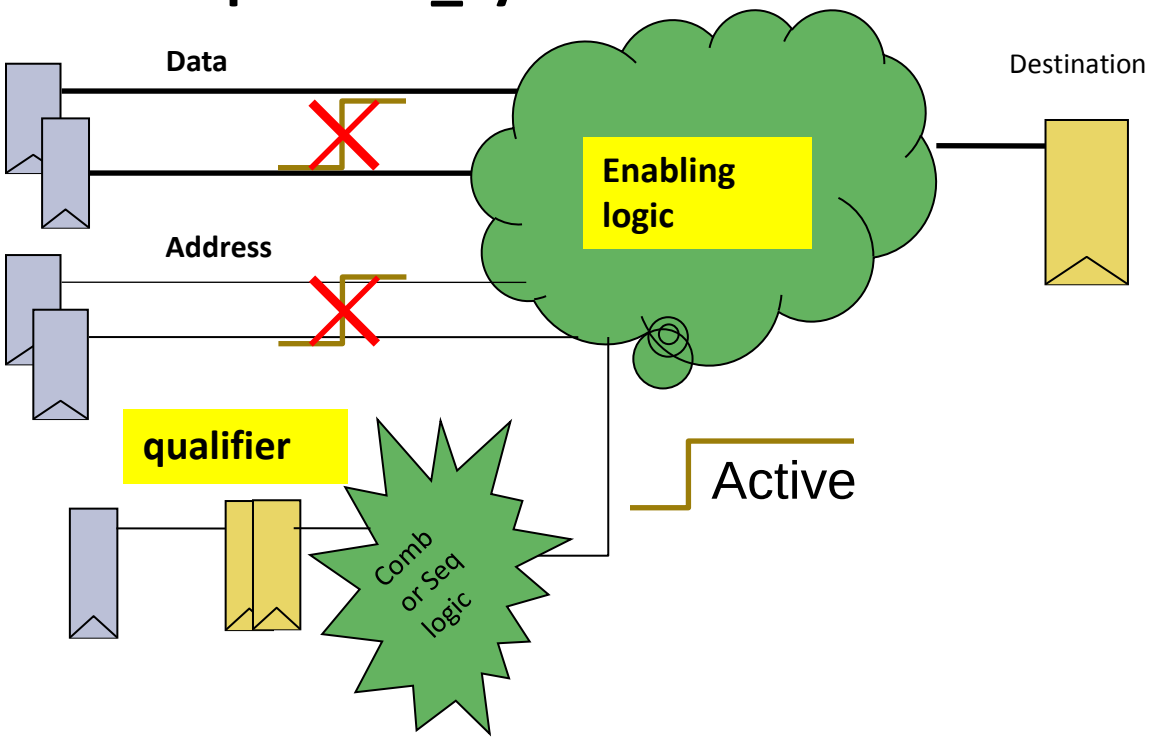
What if you do not run functional verification

- Structural CDC verification typically finds most of CDC issues, however
- You may still have some apparently properly built synchronizers that can cause metastability – e.g., qualifier not filling its duty of halting asynchronous data transfer

Atrenta Confidential © 2014 Atrenta Inc.

Ac_datahold01a: Why functional check for Ac_sync?

- **Basic concept of Ac_sync**

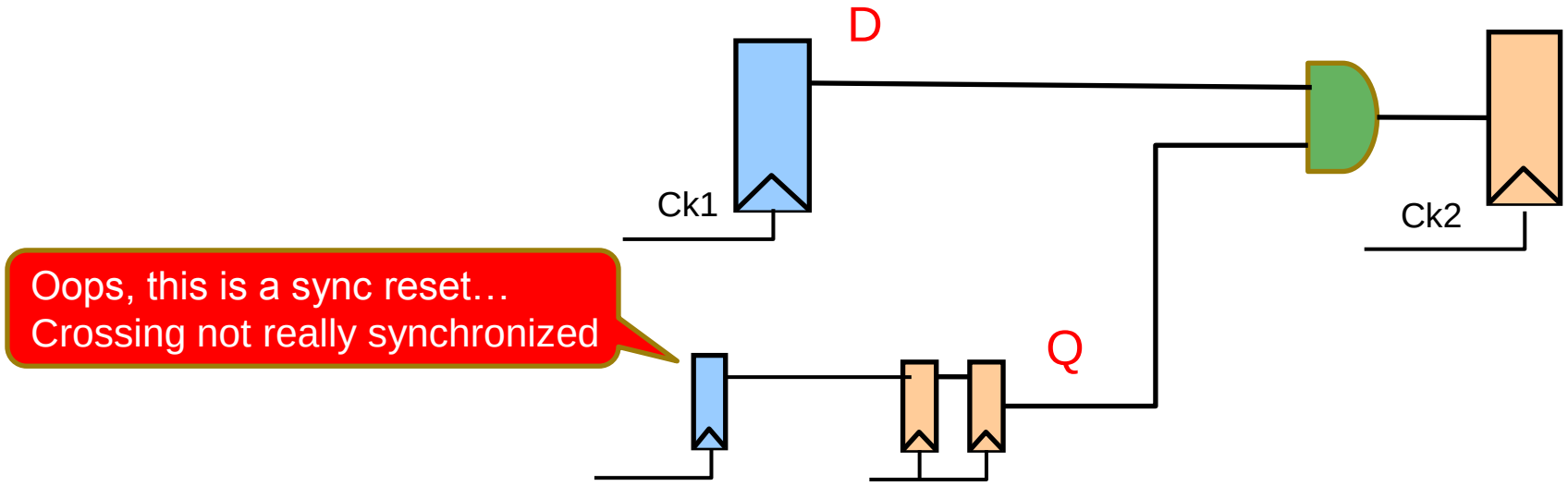


- **Structural rule *cannot* guarantee that when qualifier enables data crossing:**
  - Data will remain stable
  - Tool-detected qualifier is appropriate (for example, synchronous reset)

Atrenta Confidential © 2014 Atrenta Inc.

Data-hold check: Ac_datahold01a

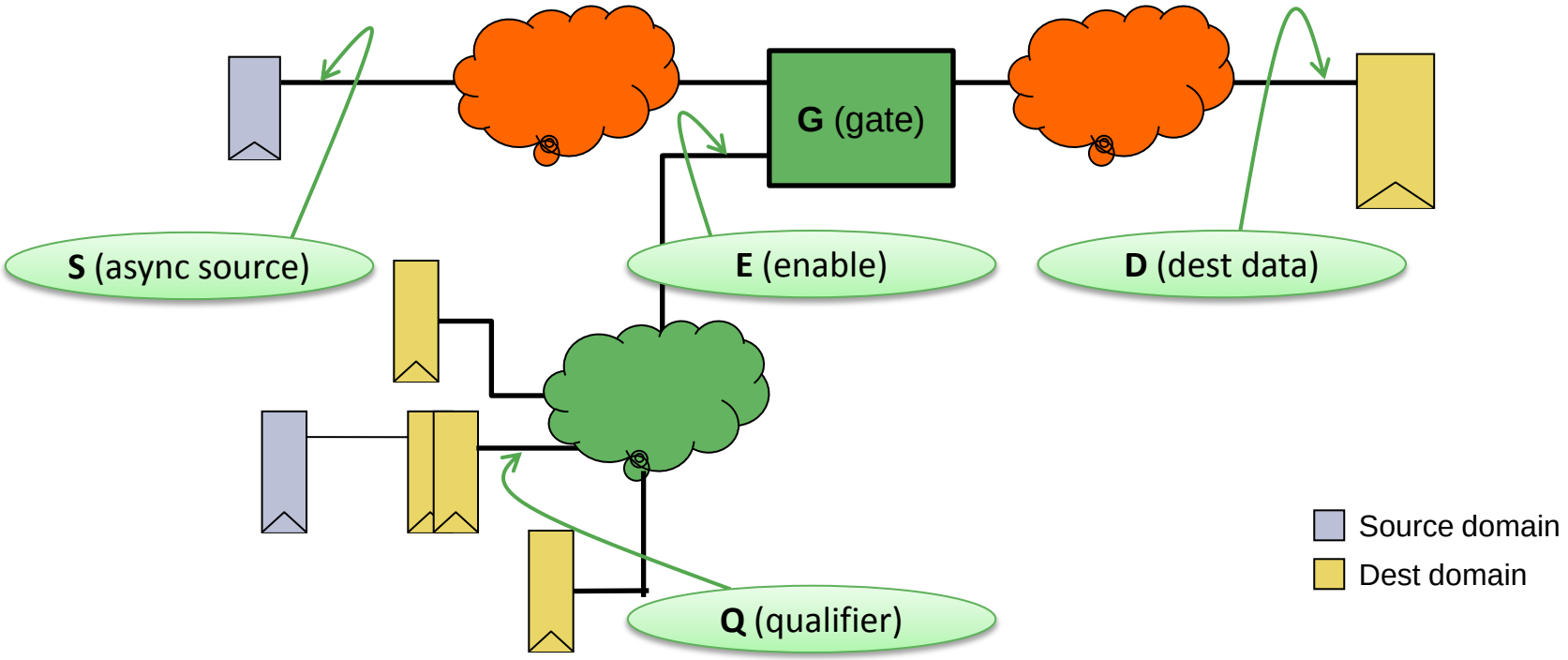
- **Properly synchronized crossing…**



- **Ac_datahold01 will ensure that if Q is asserted then D will not toggle!**
  - A synchronous reset, or any other signal not intended for synchronization will not be able to fulfill "qualifier" functional requirement and will be flagged by Ac_datahold01

Atrenta Confidential © 2014 Atrenta Inc.

Quiz: What should Ac_datahold01a check?



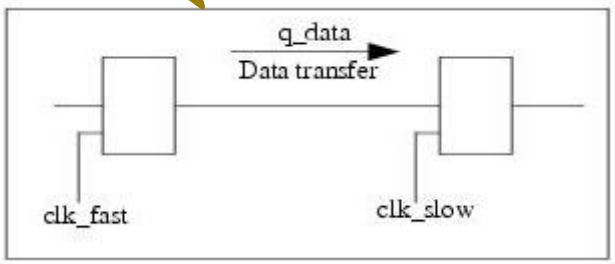
- **Should it check if S is changing when Q is active?**
  - No, because Q is not directly enabling gate G
- **Should it check if S is changing when E is active?**
  - No, because neither ensures a change at destination
- **Should it check if S is changing causing a change in D when E is active?**
  - Yes, this is what Ac_datahold01a checks

Atrenta Confidential © 2014 Atrenta Inc.

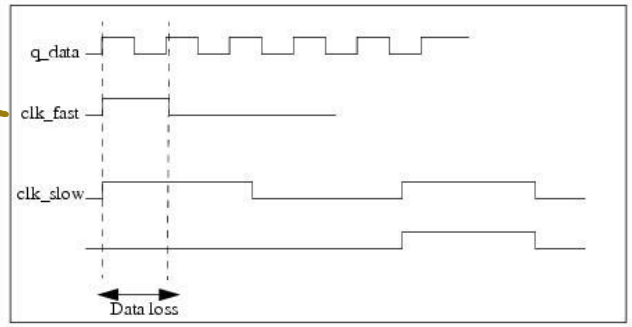
Data Loss issues: Ac_cdc01a

- **There can be data loss when**
  - There is a fast-to-slow crossings
  - Even clocks with same period defined in SGDC file, can lead to data loss issues

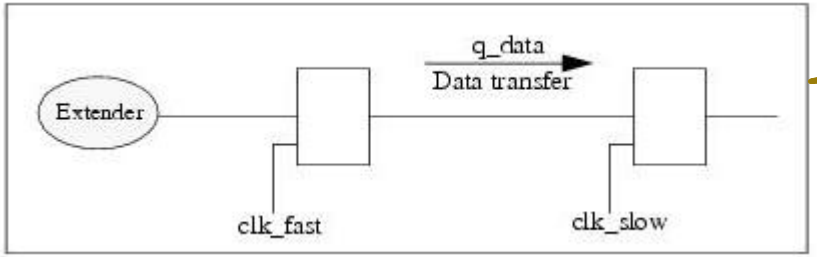
In the figure below, data passes from fast-to-slow clock domain crossing



There is a high possibility of data loss, as shown in the timing diagram



Ac_cdc01a Flags for such situations as described above



The solution is to use an extender source data for at least a full cycle of the destination clock

Atrenta Confidential © 2014 Atrenta Inc.

Rules for Identifying Converging Signals

SPYGLASS® FROM ATRENTA

- **Ac_conv02 – structural (cdc/cdc_verify_struct)**
  - Reports synchronized CDC signals (sources from single domain) that converge immediately after synchronizer
- **Ac_conv02 – functional (cdc/cdc_verify)**
  - Detects convergences of synchronized CDC signals as in structural
  - Additionally, verifies that sources of convergences are gray-encoded
  - Reports violating convergences
- **Ac_conv01 – structural-only (cdc/cdc_verify_struct)**
  - Reports synchronized CDC signals (sources from single domain) that converge after some additional sequential depth
- **Ac_conv03 – structural-only (cdc/cdc_verify_struct)**
  - Reports synchronized CDC signals (sources from *multiple* domains) that converge
- **What about signals that do not converge in current design but need to be coherent ?**
  - Example →control bus values read in software
  - Add 'gray_signals' on sources and check gray-encoding with rule Ac_conv04

Atrenta Confidential © 2014 Atrenta Inc.

Hands-on Exercise: Lab 4

- **Hands-on exercise objectives:**
  - Lab4: CDC functional verification
    - ✓ Use SpyGlass to identify clock domain crossings that cause which can lead to functional problems in the design
    - ✓ Use cdc_verify goal
    - ✓ Review and fix data coherency and combinational convergence issues
    - ✓ How to deal with partially analyzed results

- **Duration:** 60 minutes

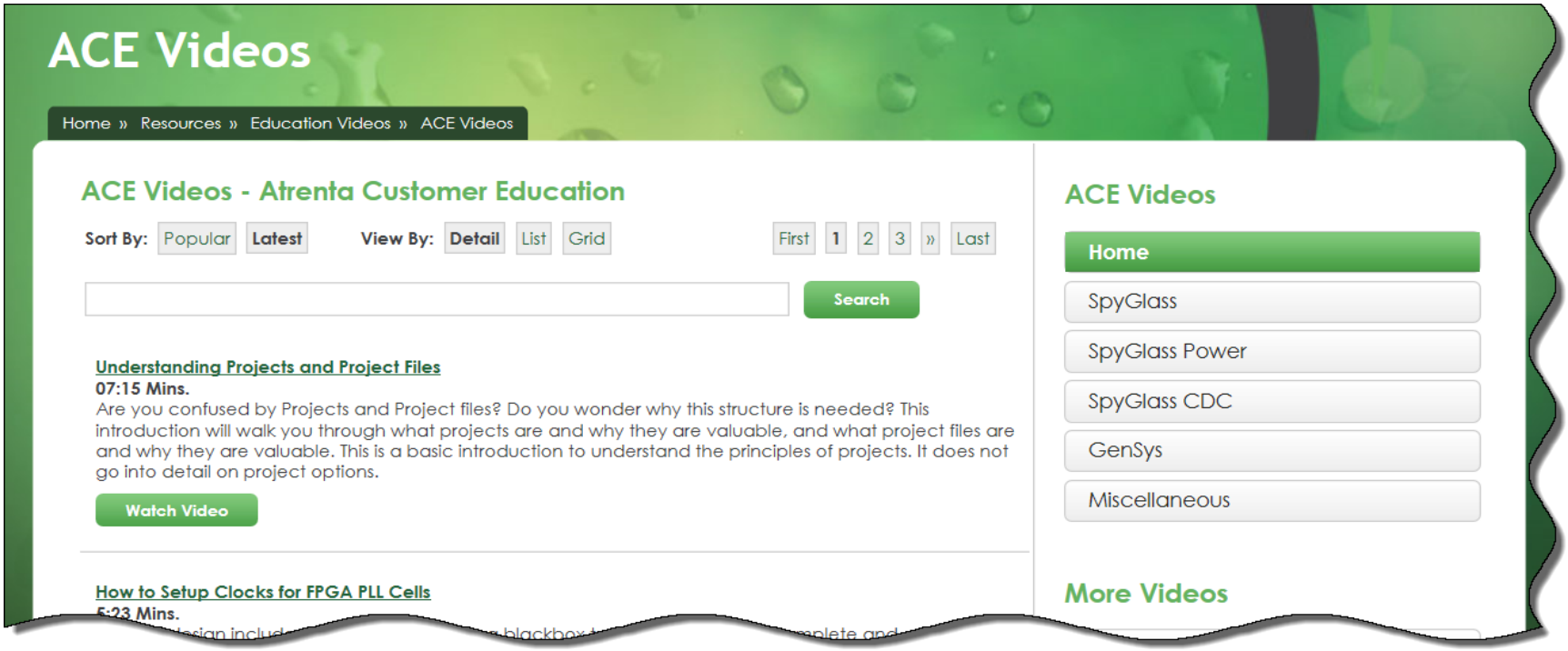
Atrenta Confidential © 2014 Atrenta Inc.

Where to Look for More Information

This presentation may contain forward-looking statements regarding product development. Information or statements contained in this presentation are for informational purposes only and do not represent a commitment, promise, or legal obligation of any kind by Atrenta Inc. or its affiliates.

Where to Look for Help → ACE Videos

- **3 – 5 Minute 'How To' Educational Videos**
  - Capabilities, Features and Best Practices
  - Searchable, Sortable, Quick and Easy
- **Atrenta.com/ACE**
  - Atrenta.com → Resources → ACE Videos



Atrenta Confidential © 2014 Atrenta Inc.

‘man’ Pages Available in ‘sg_shell’

SPYGLASS® FROM ATRENTA

- Access the ‘man’ pages from within ‘sg_shell’

```
sg_shell> man mb_connect_net

mb_connect_net(1)      sg_shell attribute documentation     mb_connect_net(1)

NAME
       mb_connect_net
           connect nets in the design

SCOPE
       mbist

SYNTAX
       mb_connect_net [-from <pin_name>] [-to <pin_name>]
                      [-tie_extra_receivers]
                      [-broadcast | -parallel | -bitwise_broadcast]
                      [-through_in_pin <pin_name>] [-through_out_pin <pin_name>]
                      [-multiplex]
                      [-select_enable <pin_name>]
                      [-active <polarity>]
                      [-cell <cell>]
                      [-add_port <bool>]
                      [-id <id_string>]

DESCRIPTION
:
```

Atrenta Confidential © 2014 Atrenta Inc.

Reference Documents

- **SpyGlass CDC Methodology User Guide**
  - Tcl CDC methodology document
  - *$SPYGLASS_HOME/doc/methodology_guides/SpyGlass_CDCMethodology_GuideWare2.0_UserGuide.pdf*
- **SpyGlass CDC Reference User Guide**
  - *$SPYGLASS_HOME/doc/policy_guides/SpyGlass_ClockResetRules_Reference.pdf*
- **SpyGlass Tcl Shell Interface**
  - Describes the interactive SpyGlass Tcl shell interface and all the available Tcl commands
  - *$SPYGLASS_HOME/doc/user_guides/SpyGlassTclShellInterface_UserGuide.pdf*

Atrenta Confidential © 2014 Atrenta Inc.

Release Information

- **The following files can be found at the top-level of install directory:**

SpyGlass_ReadMe.pdf

- Download and installation information

SpyGlass_ReleaseNotes.pdf

- What's new in this release. Also contains the list of incidents Fixed in this release

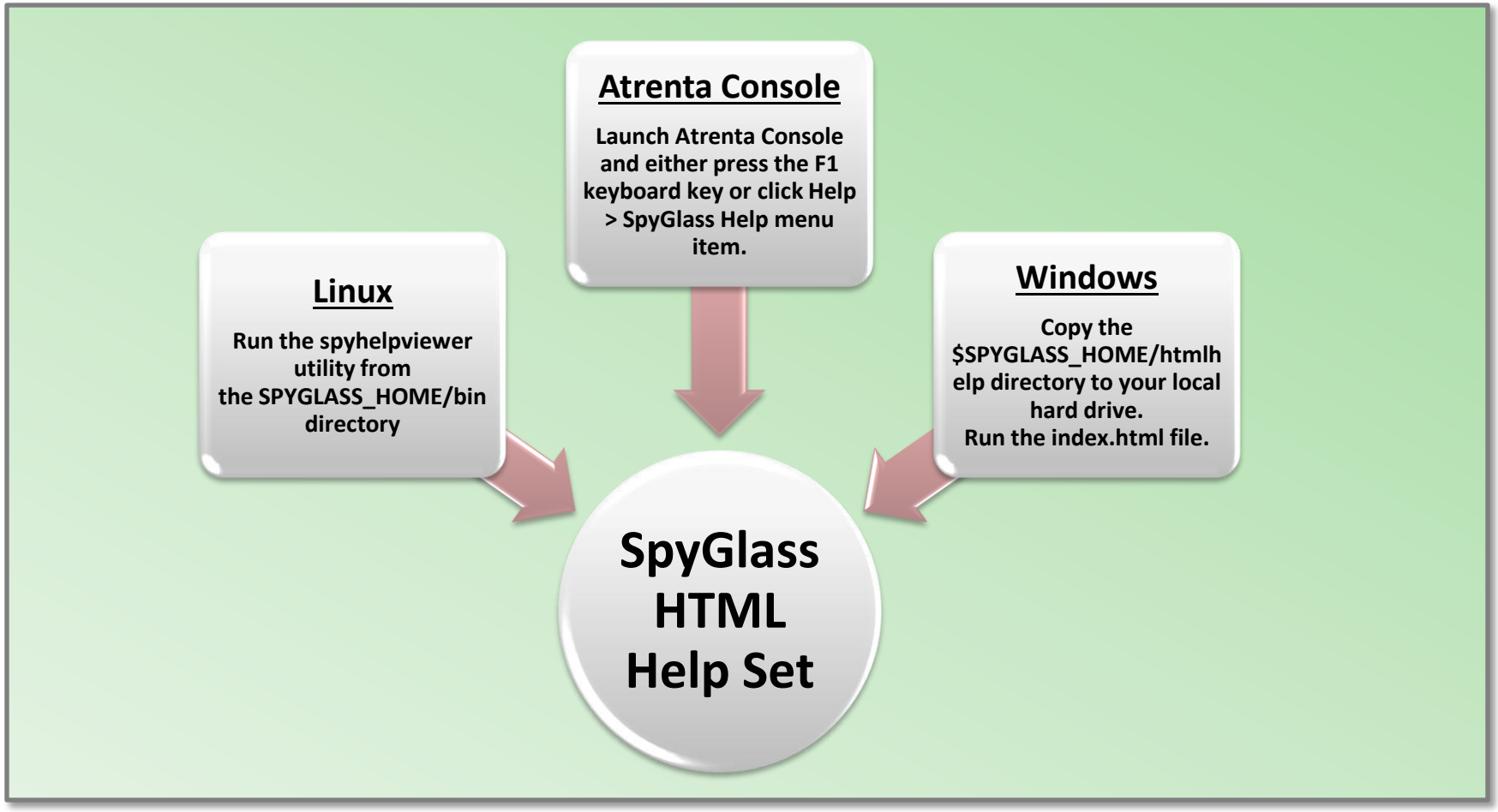
SpyGlass_KPNS.pdf

- SpyGlass_KPNS.pdf: Known problems in this release and workarounds

Atrenta Confidential © 2014 Atrenta Inc.

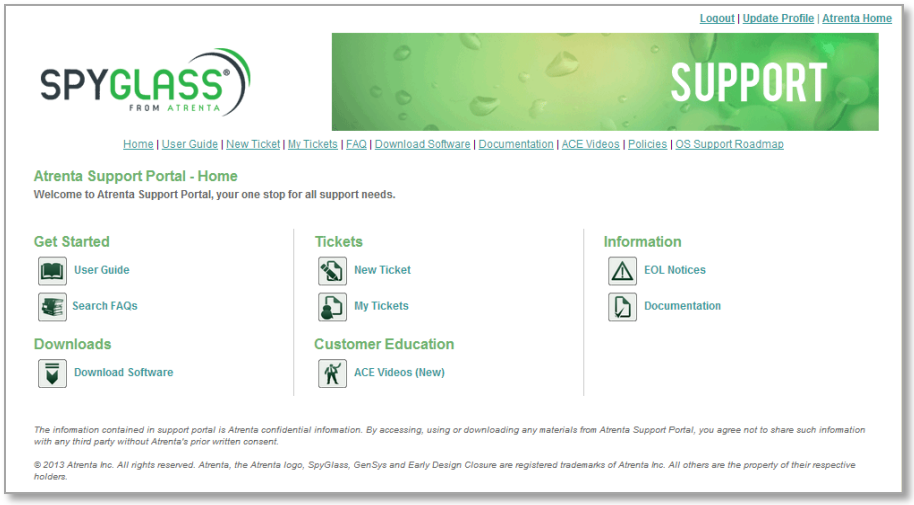
Accessing the SpyGlass HTML Help Set

- You do not have to launch Atrenta Console or run SpyGlass to access the HTML help set



Atrenta Confidential © 2014 Atrenta Inc.

Technical Support



- **Atrenta Support Portal**
  - www.atrenta.com/support
  - Provides access to the following;
  - Support system - to submit, view status and resolve support tickets
  - Download the latest SpyGlass release, release notes and What's New document
  - Product FAQ
  - And much more...

- **E-mail**
  - support@atrenta.com
  - Sending email to this address automatically creates a new ticket in the Atrenta support system. View and update the ticket via the Support Portal

Thank you!

This presentation may contain forward-looking statements regarding product development. Information or statements contained in this presentation are for informational purposes only and do not represent a commitment, promise, or legal obligation of any kind by Atrenta Inc. or its affiliates.