CS5120 VLSI System Design, Spring 2024

Dummies Guide

for SpyGlass Linter

黃稚存

Chih-Tsun Huang

cthuang@cs.nthu.edu.tw

國立清華大學
NATIONAL TSING HUA UNIVERSITY

資訊工程學系
Computer Science

Lecture 11

聲明

- 本課程之內容 (包括但不限於教材、影片、圖片、檔案資料等)，僅供修課學生個人合理使用，非經授課教師同意，不得以任何形式轉載、重製、散布、公開播送、出版或發行本影片內容(例如將課程內容放置公開平台上，如 Facebook, Instagram, YouTube, Twitter, Google Drive, Dropbox 等等)。如有侵權行為，需自負法律責任。

Outline

- Introduction
- A Quick SpyGlass Tutorial

Introduction

Lint Tool

- Lint tool, or linter, is a static code analysis tool to ensure the quality of coding style
  - To flag programming errors, bugs, stylistic errors and suspicious constructs
  - A handy tool for a software developer. You should take advantage of it!!
  - Refer to https://en.wikipedia.org/wiki/Lint_(software) if you are not aware of this kind of tools before
- For Verilog developer
  - Synopsys's SpyGlass
    - https://www.synopsys.com/verification/static-and-formal-verification/spyglass/spyglass-lint.html
  - Synopsys's nLint
    - A very nice lint tool. Unfortunately, Synopsys phased out this tool and replaced with SpyGlass

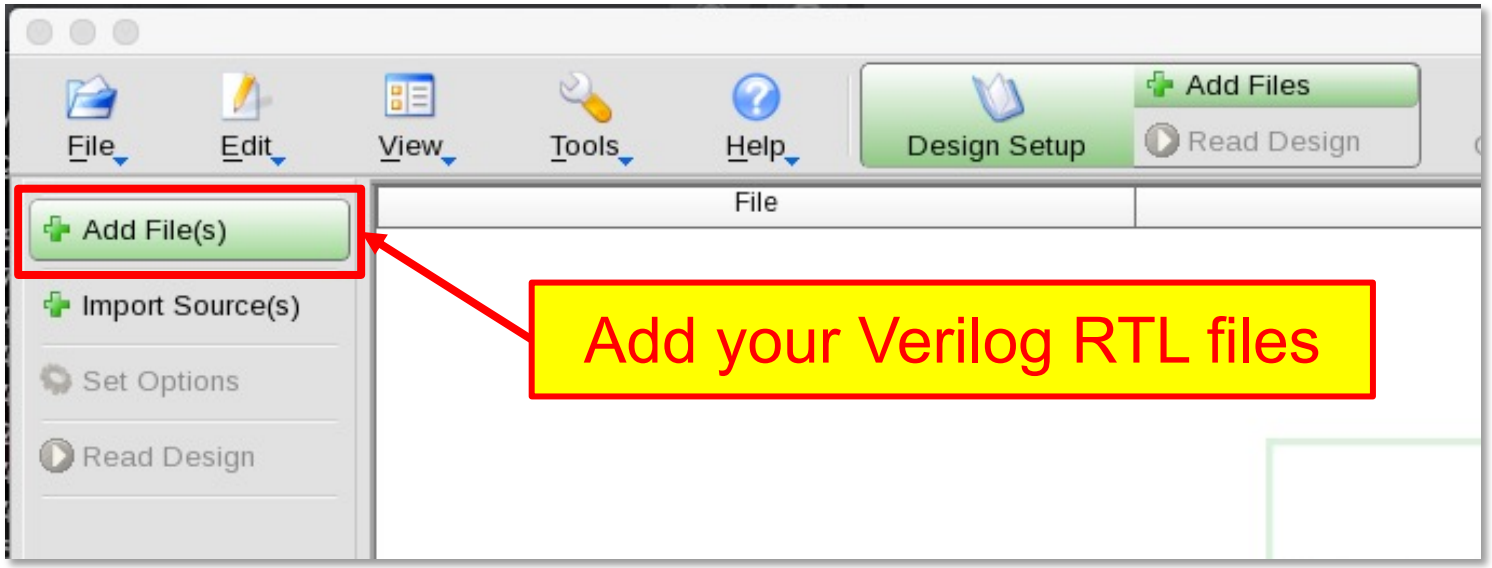
A Quick SpyGlass Tutorial

Run SpyGlass and Add Verilog Files

- You should check which servers can execute SpyGlass first
- Login with `ssh -X` (or `-Y` for MacOS) and type

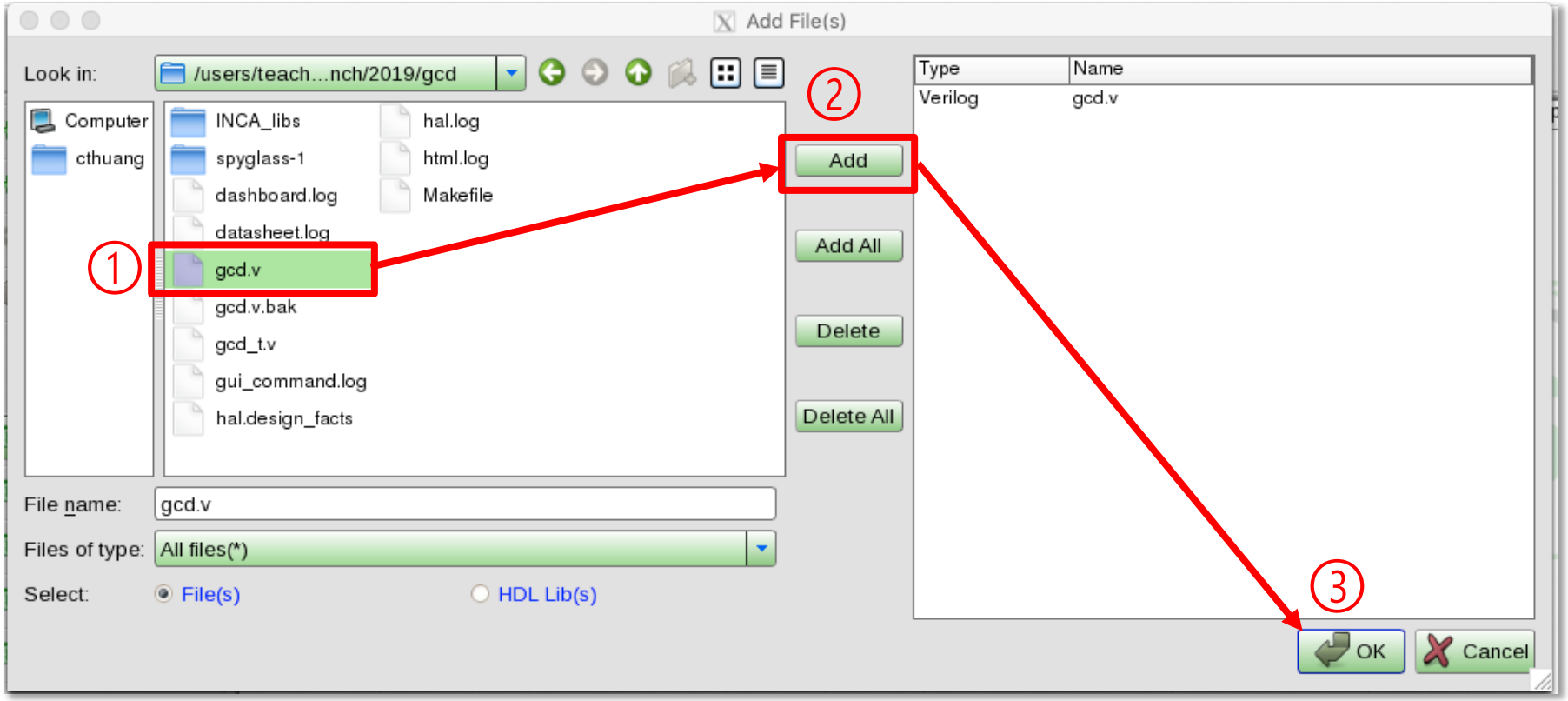
```
$ source /usr/cad/synopsys/CIC/spyglass.cshrc
$ spyglass &
```

Only need to source it once for each log-in

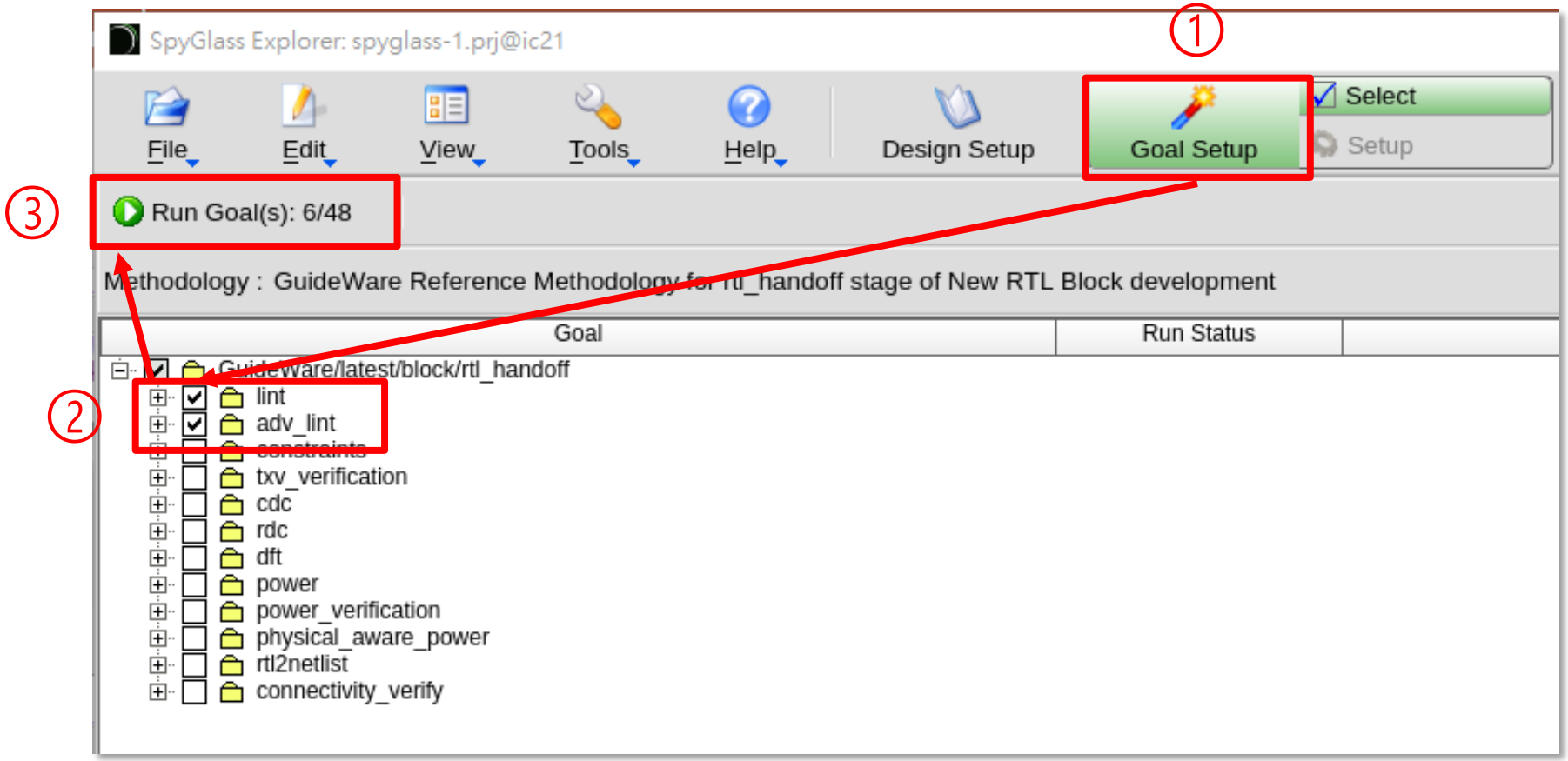


Check the RTL Files

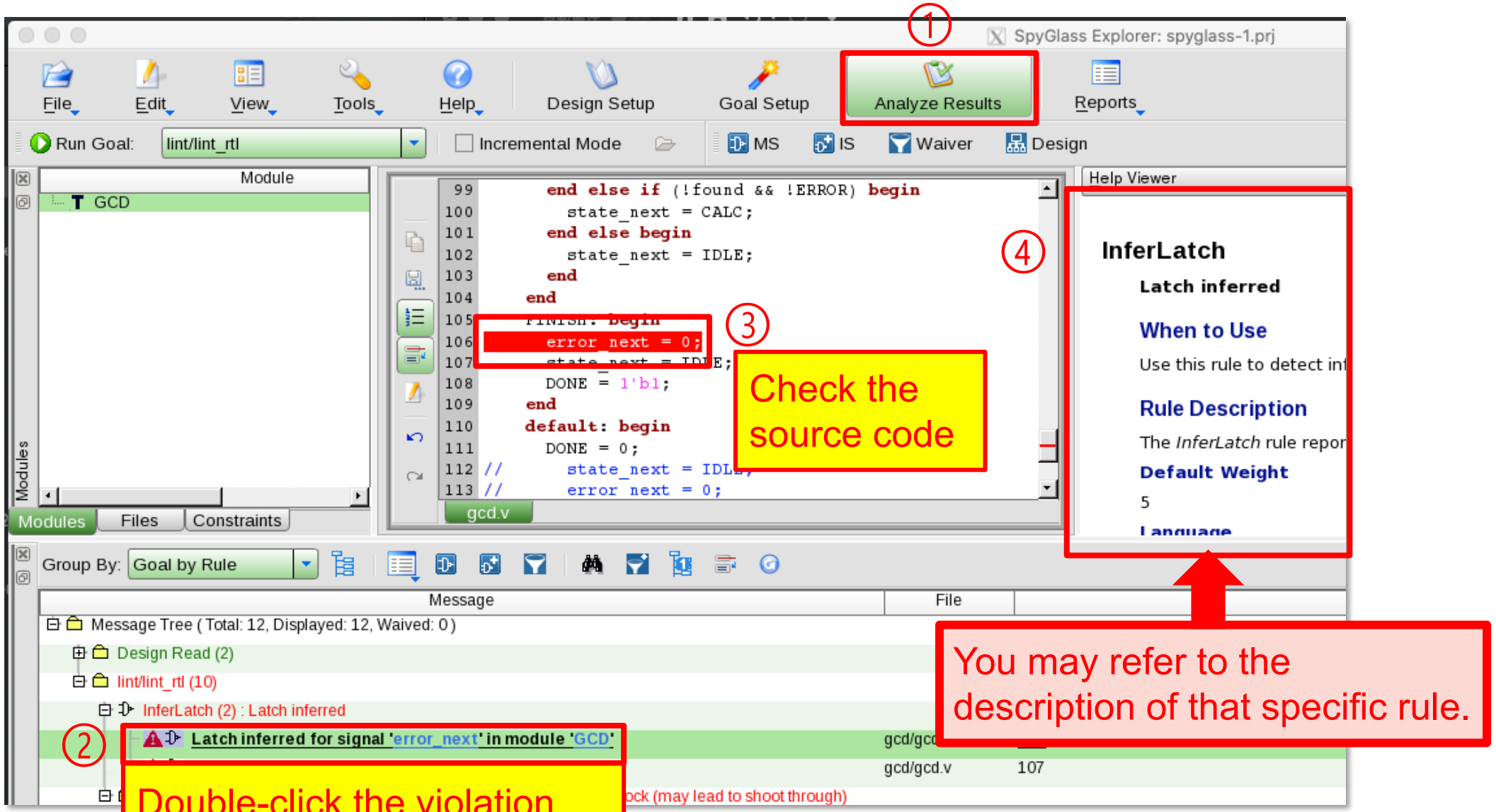
- Remember that NO test stimulus for lint check



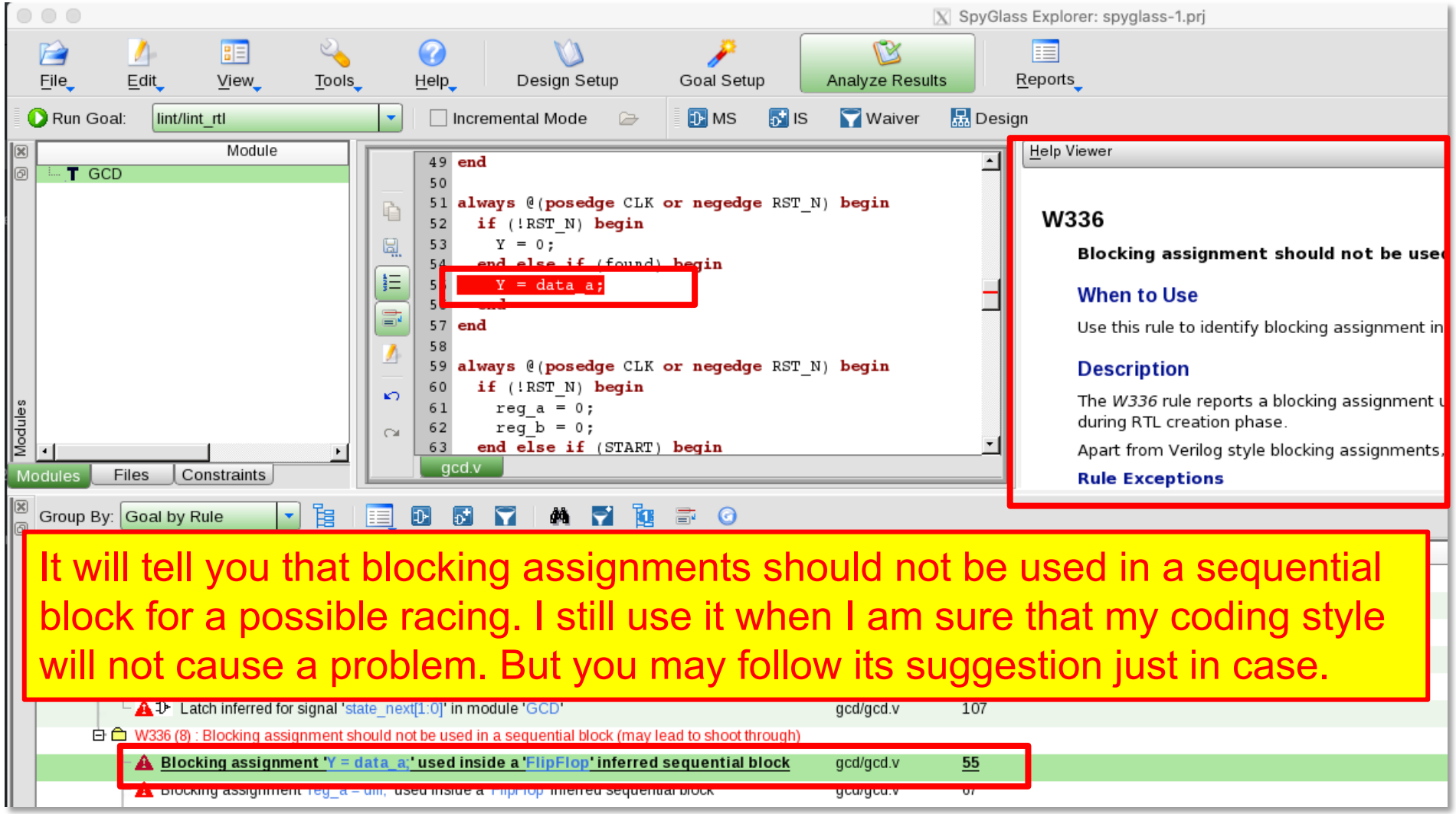
Select the Lint Goal and Run



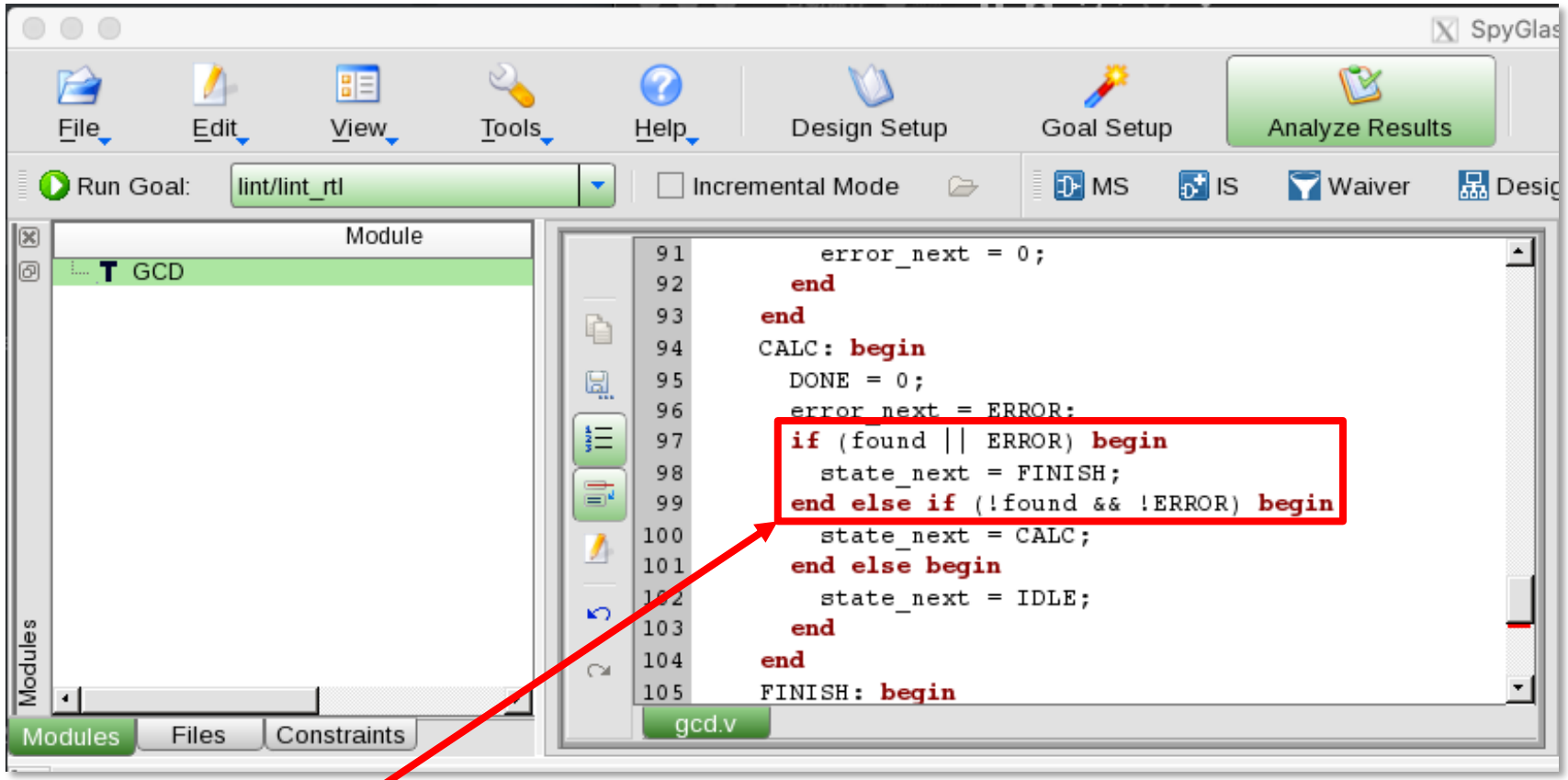
Analyze Results (Example 1)



Analyze Results (Example 2)



Analyze Results (Example 3)



Unfortunately, SpyGlass cannot check this kind of redundancy that we discussed in the lecture note. You may still use the code coverage tool to check the design redundancy.

Some Linter Can Identify Specific Redundant Code

- nLint can point out the following code redundancy
- Unfortunately, SpyGlass is not the one…
- You may still use a code coverage tool to check it

```
CALC: begin
        error_next = ERROR;
        if (found || ERROR) begin
          state_next = FINISH;
        end else if (!found && !ERROR) begin
          state_next = CALC;
        end
      end
```

Redundant!

Rule of Thumb

- Synthesizable code
- Two-process FSM style
- Coding elegance
  - Consistent naming convention
  - Consistent indention
- Proper commenting

→ Following the coding styles I recommended!

Lint is only a tool to help your style!

- Avoid all errors and warnings
  - Otherwise make sure you can explain them
- Read the description by the tool to identify any bad style!
  - False alerts?
  - What is good and what is bad?
  - Define your own rule set
- Some BAD coding style can be synthesized as well
  - RTL Synthesizer (e.g., Design Compiler) can be a rule checker (coding style checker) too.
  - Use

    ```
    dc_shell> check_design
    ```

    after reading your Verilog code in dc_shell

Command-Line Script to Generate SpyGlass Report

spyglass.tcl

```
# readfile
set HDL_DIR "../hdl"
set SOURCE_FILE "$HDL_DIR/lenet.v"
set RPT_FILE "spyglass.rpt"
read_file -type verilog $SOURCE_FILE

#goal setup (lint_rtl)
current_goal lint/lint_rtl -alltop
run_goal
capture $RPT_FILE {write_report moresimple}

#goal setup (lint_turbo_rtl)
current_goal lint/lint_turbo_rtl -alltop
run_goal
capture -append $RPT_FILE {write_report moresimple}

#goal setup (lint_functional_rtl)
current_goal lint/lint_functional_rtl -alltop
run_goal
capture -append $RPT_FILE {write_report moresimple}
```

```
#goal setup (lint_abstract)
current_goal lint/lint_abstract -alltop
run_goal
capture -append $RPT_FILE {write_report moresimple}

#goal setup (adv_lint_struct)
current_goal adv_lint/adv_lint_struct -alltop
run_goal
capture -append $RPT_FILE {write_report moresimple}

#goal setup (adv_lint_verify)
current_goal adv_lint/adv_lint_verify -alltop
run_goal
capture -append $RPT_FILE {write_report moresimple}
```

Command-Line Script to Generate SpyGlass Report

- Edit `spyglass.tcl`
  - Modify HDL_DIR, SOURCE_FILE, and RPT_FILE to match your environment
  - Try to understand the script
    - You may refer to
      /usr/cad/synopsys/spyglass/cur/SPYGLASS_HOME/doc/user_guides
      for Tcl Shell Interface User Guide and Built-in Rules Reference Guide
- Execute the script in the terminal and enjoy the report

```
$  sg_shell < spyglass.tcl
```