

# SystemVerilog Configurations and Tool Flow Using SCons (an Improved Make)

Don Mills

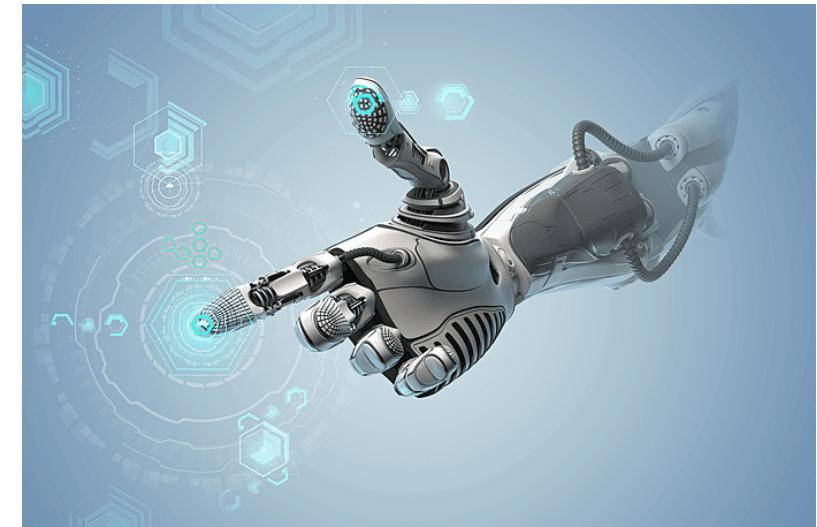
Microchip Technology Inc.

Dillan Mills

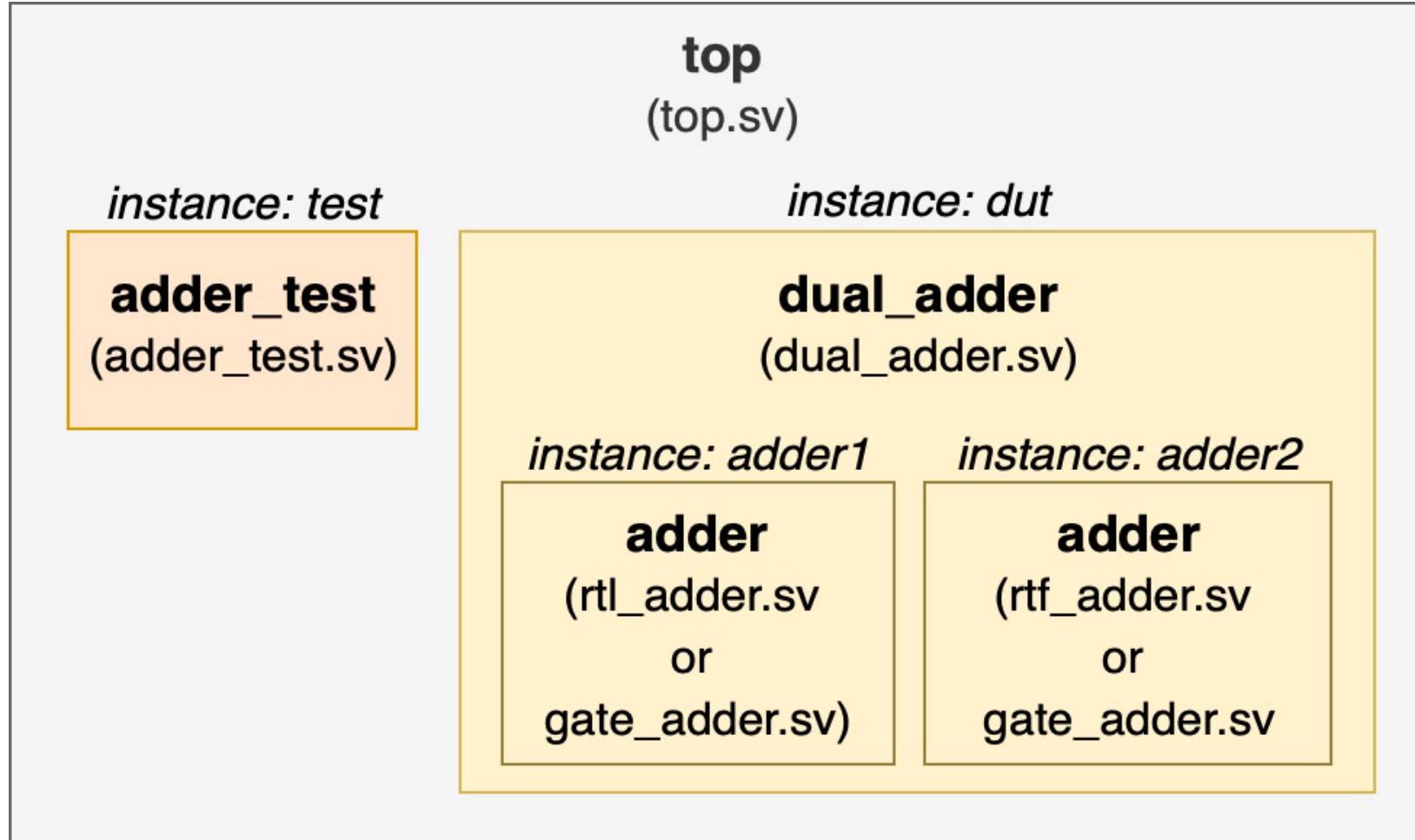
Microchip Technology Inc.

# Motivation for paper

- SystemVerilog Configurations
  - No one knows how to set up configs for the tools
    - (except the experts)
- SCons
  - Like Make but...
  - better - faster - stronger
    - We can build it, “we have the technology”
- Why combine both in one paper?



# Test Circuit for Paper



# Configuration Libraries



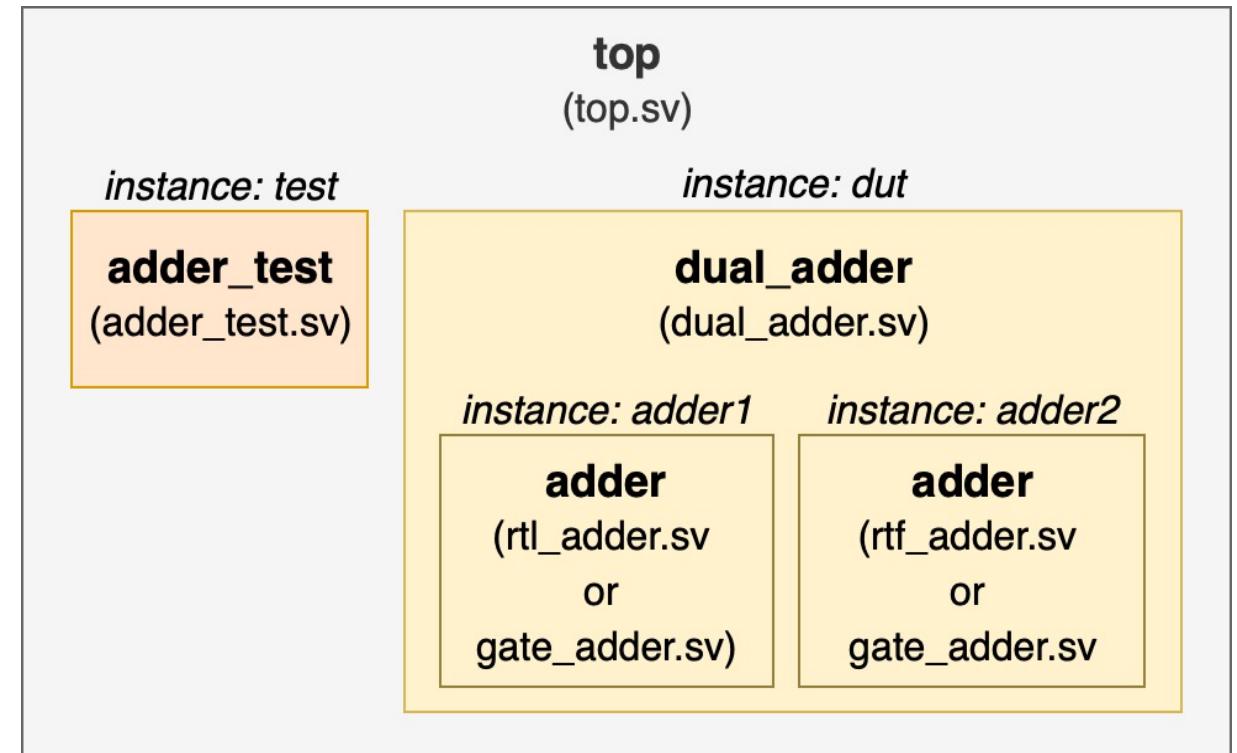
- Two parts to Configurations
  - Library files
    - Declares which files are in which library
  - Config files (next slide)

## libmap.sv file

```
library rtlLib  rtl_adder.sv,
              dual_adder.sv;

library gateLib gate_adder.sv,
                gate_adder_alt.sv;

library testLib top.sv,
               adder_test.sv;
```



# Configuration Files

- Two parts to Configurations (Continued)
  - Config files
    - Declares the design top and which library is resides in



## config.sv file

```
config rtl_config;  
  design testLib.top;  
  default liblist rtlLib;  
endconfig  
  
config cell_config;  
  design testLib.top;  
  default liblist rtlLib;  
  cell adder use gateLib.adder;  
endconfig  
  
config inst_config;  
  design testLib.top;  
  default liblist rtlLib;  
  instance top.dut.adder2  
    use gateLib.adder_alt;  
endconfig
```

# Configuration Files

- Two parts to Configurations (Continued)
  - Config files
    - Declares the design top and which library is resides in
    - Declares a default library for searching cells not otherwise specified



## config.sv file

```
config rtl_config;  
  design testLib.top;  
  default liblist rtlLib;  
endconfig  
  
config cell_config;  
  design testLib.top;  
  default liblist rtlLib;  
  cell adder use gateLib.adder;  
endconfig  
  
config inst_config;  
  design testLib.top;  
  default liblist rtlLib;  
  instance top.dut.adder2  
    use gateLib.adder_alt;  
endconfig
```

# Configuration Files

- Two parts to Configurations (Continued)
  - Config files
    - Declares the design top and which library is resides in
    - Declares a default library for searching cells not otherwise specified
    - Declares replacement item(s)
      - By cell or by instance



## config.sv file

```
config rtl_config;  
  design testLib.top;  
  default liblist rtlLib;  
endconfig  
  
config cell_config;  
  design testLib.top;  
  default liblist rtlLib;  
  cell adder use gateLib.adder;  
endconfig  
  
config inst_config;  
  design testLib.top;  
  default liblist rtlLib;  
  instance top.dut.adder2  
    use gateLib.adder_alt;  
endconfig
```

# Configuration Files

- Two parts to Configurations (Continued)
  - Config files
    - Declares the design top and which library is resides in
    - Declares a default library for searching cells not otherwise specified
    - Declares replacement item(s)
      - By cell or by instance
      - Which library to find the replacement item(s)



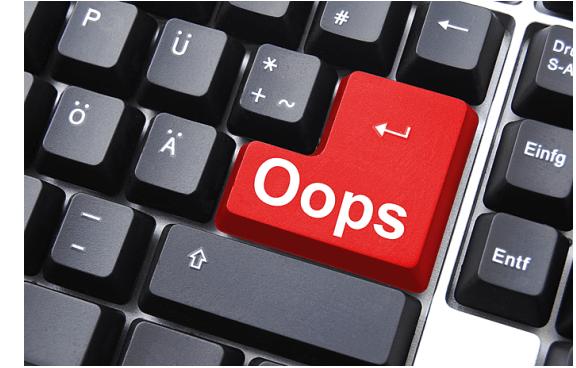
## config.sv file

```
config rtl_config;  
  design testLib.top;  
  default liblist rtlLib;  
endconfig  
  
config cell_config;  
  design testLib.top;  
  default liblist rtlLib;  
  cell adder use gateLib.adder;  
endconfig  
  
config inst_config;  
  design testLib.top;  
  default liblist rtlLib;  
  instance top.dut.adder2  
    use gateLib.adder_alt;  
endconfig
```

# Configuration File Gotcha

- The default liblist can contain a list of libraries

```
config rtl_config;
  design testLib.top;
  default liblist rtlLib testLib;
endconfig
```



- This list of libraries is one of the few (maybe the only) place in SystemVerilog where a list of items are not comma separated

# Vendor Command Switches

(alphabetically)

## Cadence

```
prompt> xrun -libmap libmap.sv \
           -compcnfg configs.sv \
           -f source.f \
           -top cell_config
```

- Specify in the xrun command
  - the libmap file



source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

# Vendor Command Switches

(alphabetically)

## Cadence

```
prompt> xrun -libmap libmap.sv \
           -compcnfg configs.sv \
           -f source.f \
           -top cell_config
```



- Specify in the xrun command
  - the libmap file
  - the configuration file

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

# Vendor Command Switches

(alphabetically)

## Cadence

```
prompt> xrun -libmap libmap.sv \
           -compcnfg configs.sv \
           -f source.f \
           -top cell_config
```

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

- Specify in the xrun command
  - the libmap file
  - the configuration file
  - which configuration you want to use from the config file



# Vendor Command Switches

(alphabetically)

## Cadence

```
prompt> xrun -libmap libmap.sv \
           -compcnfg configs.sv \
           -f source.f \
           -top cell_config
```

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

- Specify in the xrun command
  - the libmap file
  - the configuration file
  - which configuration you want to use from the config file
  - and of course, the source file



# Vendor Command Switches

(alphabetically)

## Cadence

```
prompt> xrun -libmap libmap.sv \
           -compcnfg configs.sv \
           -f source.f \
           -top cell_config
```

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

- Specify in the xrun command
  - the libmap file
  - the configuration file
  - which configuration you want to use from the config file
  - and of course, the source file

*Must be last in list for one of the vendors*



# Vendor Command Switches

(alphabetically)

## Mentor



```
prompt> vlog -libmap libmap.sv \
           -f source.f \
```

```
prompt> vsim cell_config \
           -c \
           -do run.do \
```

run.do file  
run -all  
exit

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

- Compile (vlog) the libmap and source files

# Vendor Command Switches

(alphabetically)

## Mentor

```
prompt> vlog -libmap libmap.sv \
           -f      source.f
```

```
prompt> vsim cell_config
           -c
           -do run.do
```

run.do file  
run -all  
exit

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

- Compile (vlog) the libmap and source files
- Specify in the vsim command which configuration you want to use from the config file



# Vendor Command Switches

(alphabetically)

## Mentor

```
prompt> vlog -libmap libmap.sv \
           -f          source.f
```

```
prompt> vsim cell_config \
           -c          \
           -do run.do
```

run.do file  
run -all  
exit

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

- Compile (vlog) the libmap and source files
- Specify in the vsim command which configuration you want to use from the config file

*Must be last in list for one of the vendors*



# Vendor Command Switches

(alphabetically)



## Synopsys

- Compile (vlogan) the libmap and source files

```
prompt> vlogan -full64
          -diag libconfig
          -svverilog
          -libmap libconfig
          -f source.f
```

```
prompt> vcs -full64
          -diag libconfig
          -debug_access
          -R cell_config
          -ucli
          -i run_vcs.do
```

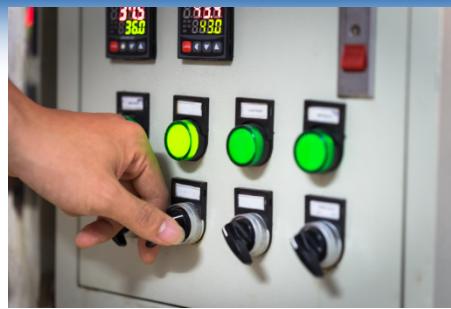
run\_vcs.do file  
run  
exit

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

synopsys\_sim.setup file  
WORK > DEFAULT  
DEFAULT : work  
rtlLib : rtlLib  
gatelib : gateLib  
testlib : testLib

# Vendor Command Switches

(alphabetically)



## Synopsys

- Compile (vlogan) the libmap and source files
- Specify in the vcs command which configuration you want to use from the config file

```
prompt> vlogan -full64
          -diag libconfig \
          -sverilog \
          -libmap libconfig \
          -f source.f
```

```
prompt> vcs -full64
          -diag libconfig \
          -debug_access \
          -R cell_config \
          -ucli
          -i run_vcs.do
```

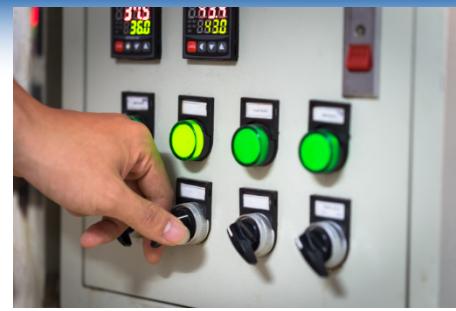
run\_vcs.do file  
run  
exit

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

synopsys\_sim.setup file  
WORK > DEFAULT  
DEFAULT : work  
rtlLib : rtlLib  
gatelib : gateLib  
testlib : testLib

# Vendor Command Switches

(alphabetically)



## Synopsys

- Compile (vlogan) the libmap and source files
- Specify in the vcs command which configuration you want to use from the config file
- The setup file must be visible

```
prompt> vlogan -full64
          -diag libconfig \
          -sverilog \
          -libmap libconfig \
          -f source.f
```

```
prompt> vcs -full64
          -diag libconfig \
          -debug_access \
          -R cell_config \
          -ucli \
          -i run_vcs.do
```

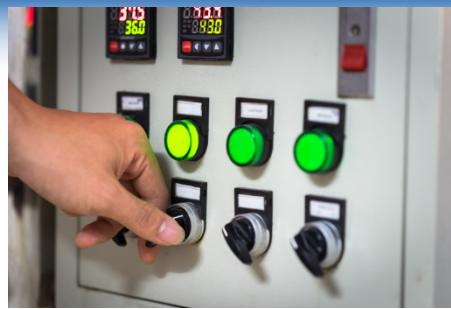
run\_vcs.do file  
run  
exit

source.f file  
 adder\_test.sv  
 dual\_adder.sv  
 gate\_adder.sv  
 gate\_adder\_alt.sv  
 rtl\_adder.sv  
 top.sv  
 configs.sv

synopsys\_sim.setup file  
 WORK > DEFAULT  
 DEFAULT : work  
 rtlLib : rtlLib  
 gatelib : gateLib  
 testlib : testLib

# Vendor Command Switches

(alphabetically)



## Synopsys

- Compile (vlogan) the libmap and source files
- Specify in the vcs command which configuration you want to use from the config file
- The setup file must be visible

```
prompt> vlogan -full64
          -diag libconfig \
          -sverilog \
          -libmap libconfig \
          -f source.f
```

```
prompt> vcs -full64
          -diag libconfig \
          -debug_access \
          -R cell_config \
          -ucli \
          -i run_vcs.do
```

run\_vcs.do file  
run  
exit

Must be last in list for one of the vendors

source.f file  
adder\_test.sv  
dual\_adder.sv  
gate\_adder.sv  
gate\_adder\_alt.sv  
rtl\_adder.sv  
top.sv  
configs.sv

synopsys\_sim.setup file  
 WORK > DEFAULT  
 DEFAULT : work  
 rtlLib : rtlLib  
 gateLib : gateLib  
 testLib : testLib

# SCons: A Software Construction Tool

Configuration files are all Python scripts

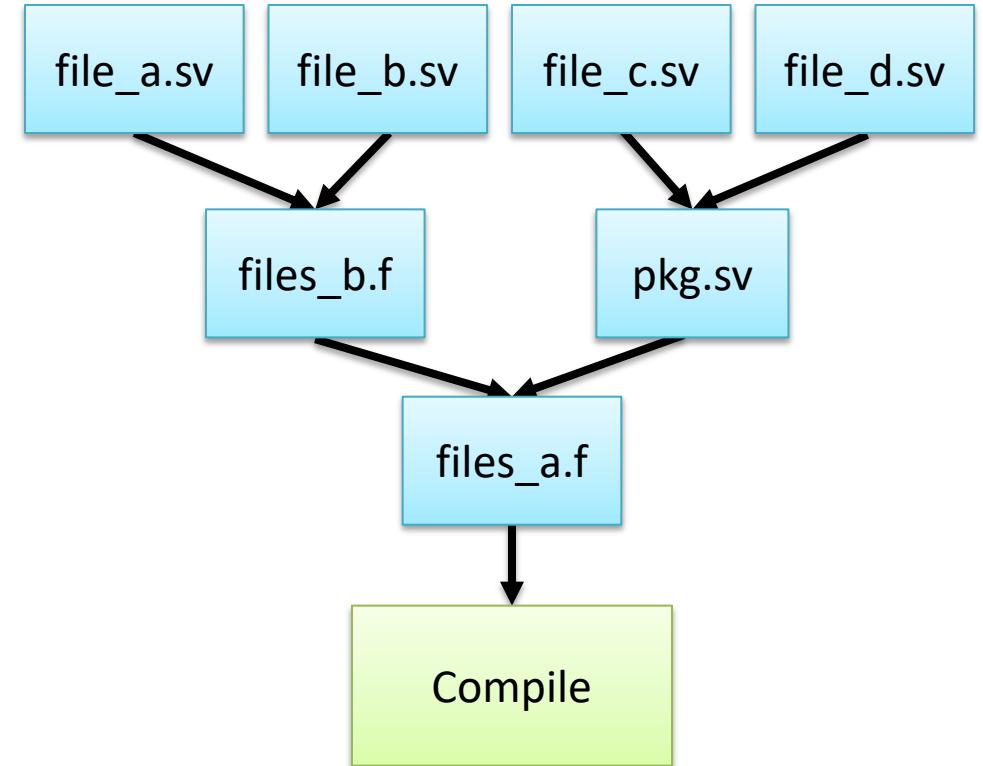
Use the power of a complete programming language to solve complex build configuration issues



# Automatic Dependency Tree Generation

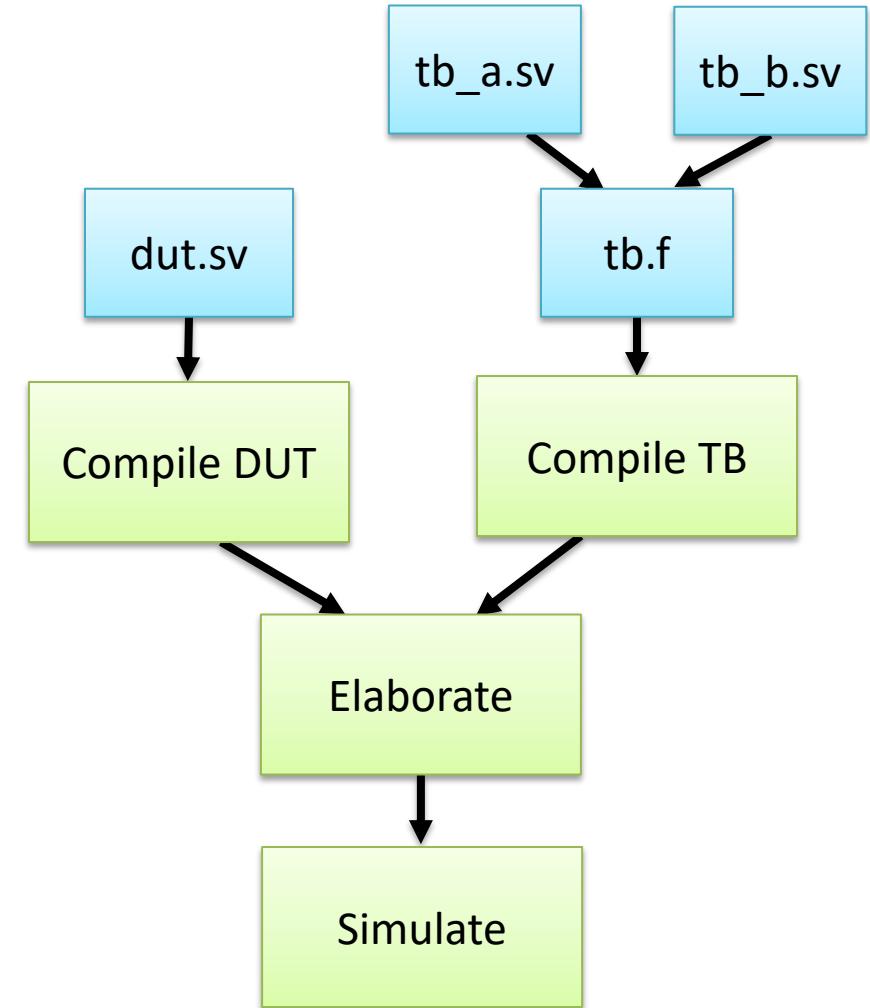
- Supports a regex-based “scanner” that searches files for dependencies
- Make SCons aware of the root file and it will monitor all files for changes
- Only re-compiles when a change is detected

No EDA tools necessary  
to create a Makefile



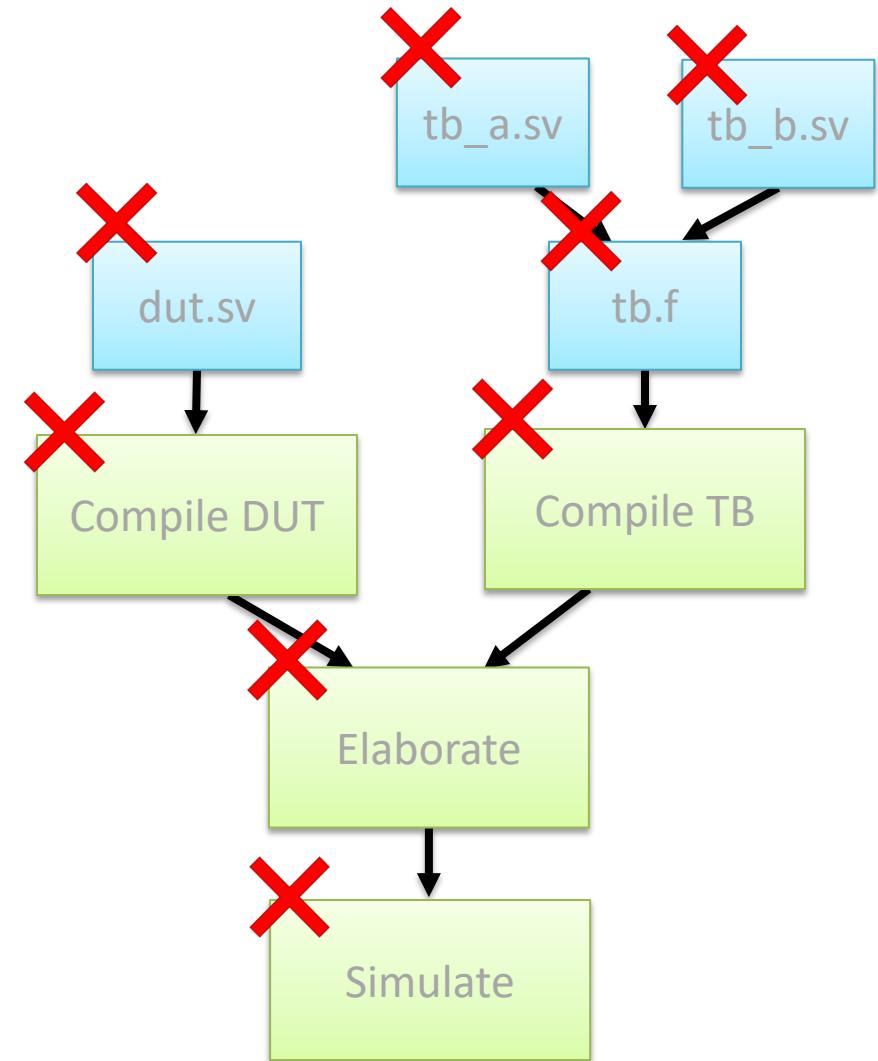
`env.Compile(compile.log, files_a.f)`

# Automatic Dependency Tracking



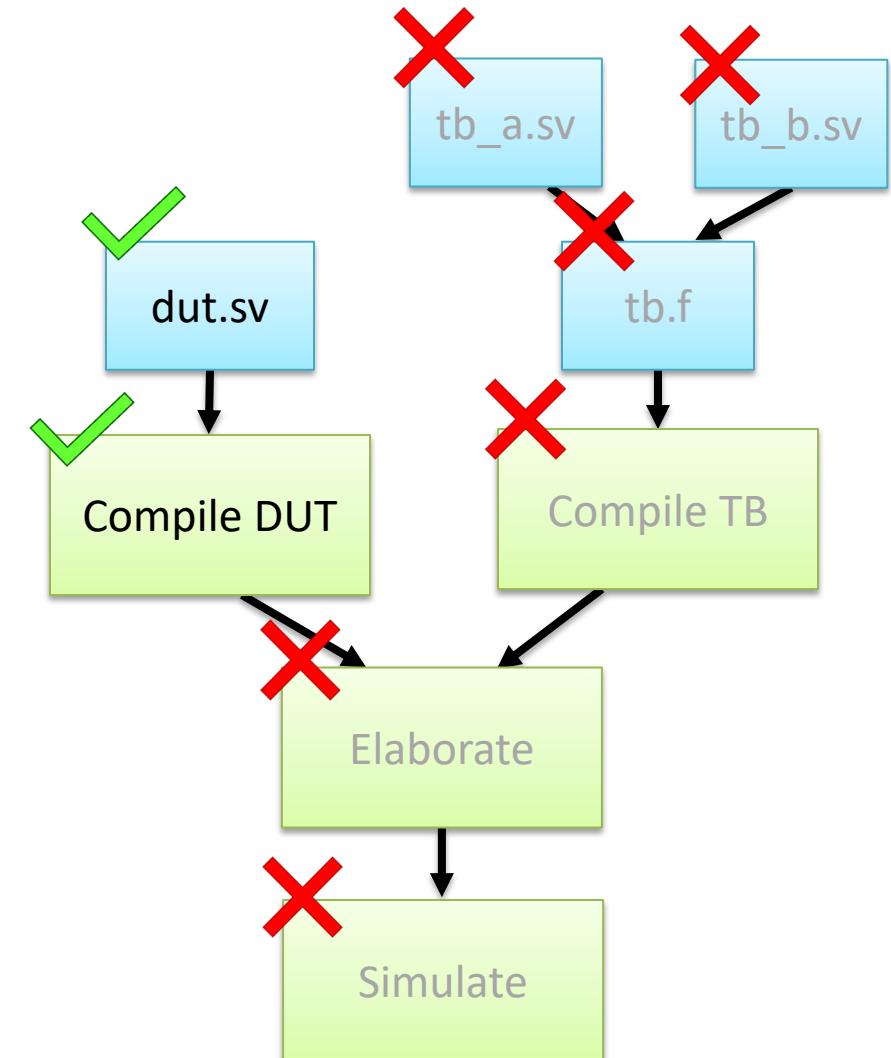
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line



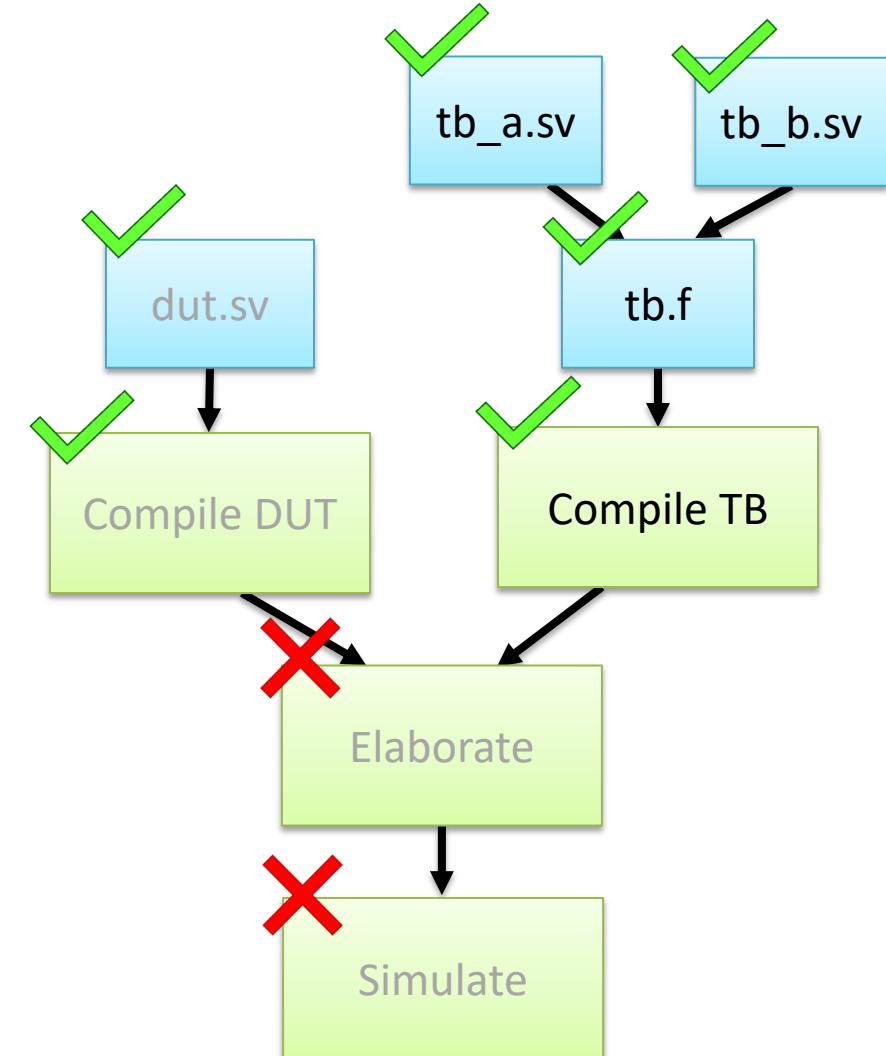
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT



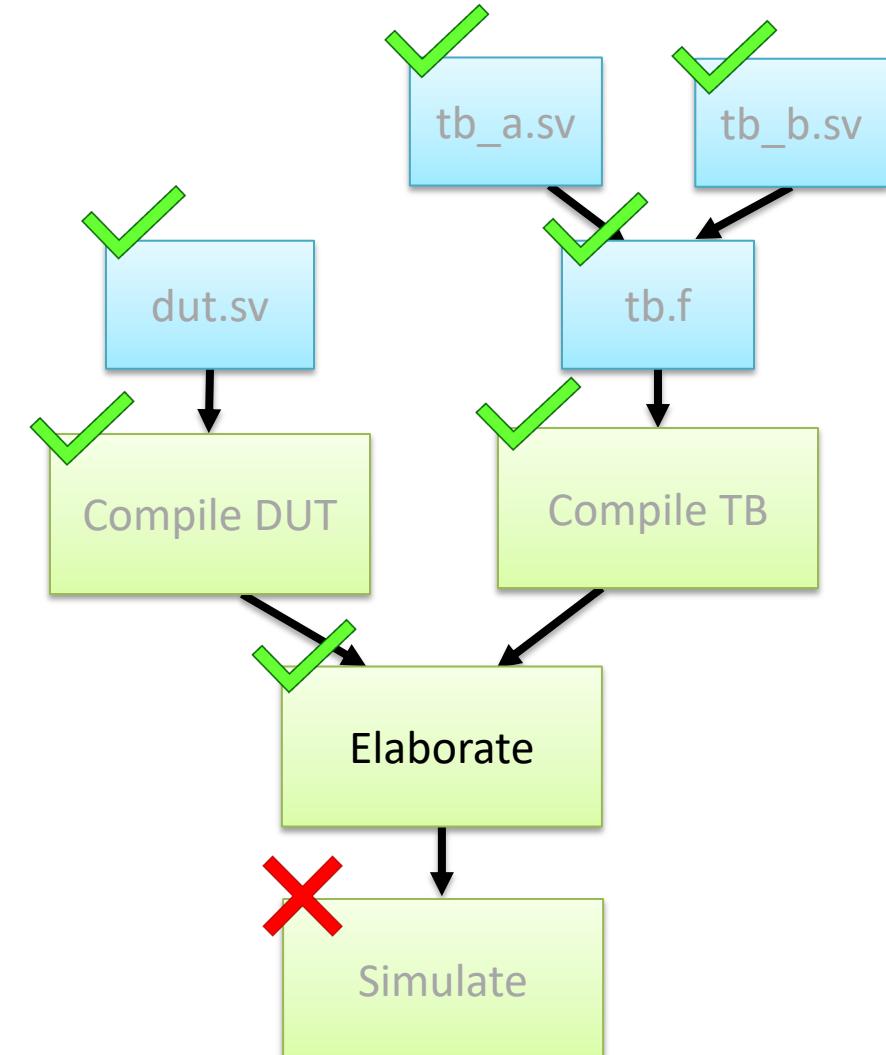
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB



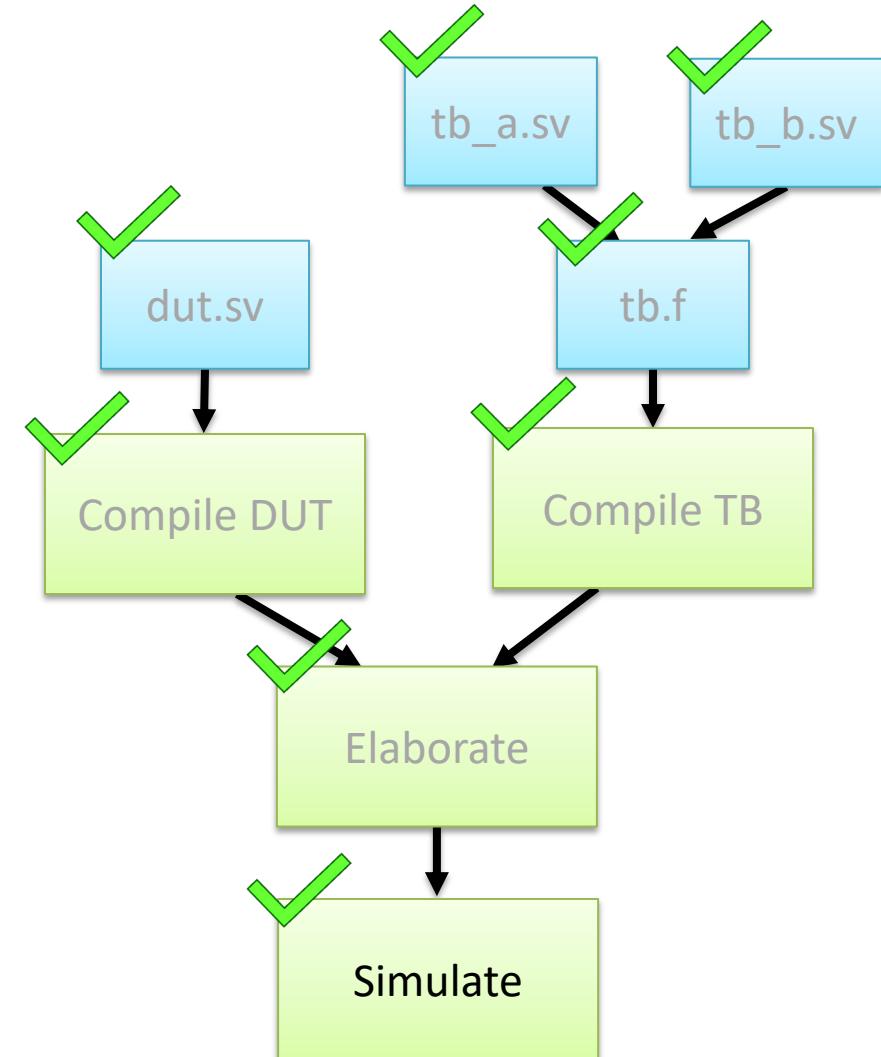
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB
  - Elaborate



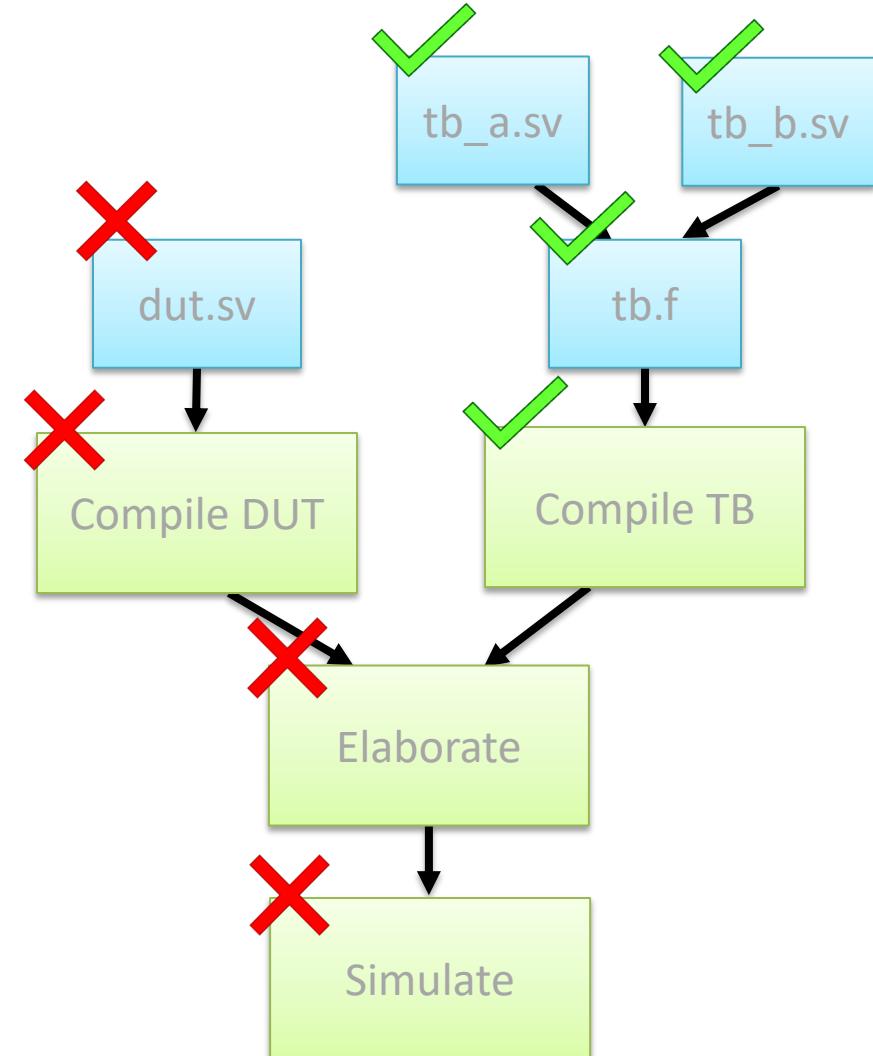
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB
  - Elaborate
  - Start a Simulation



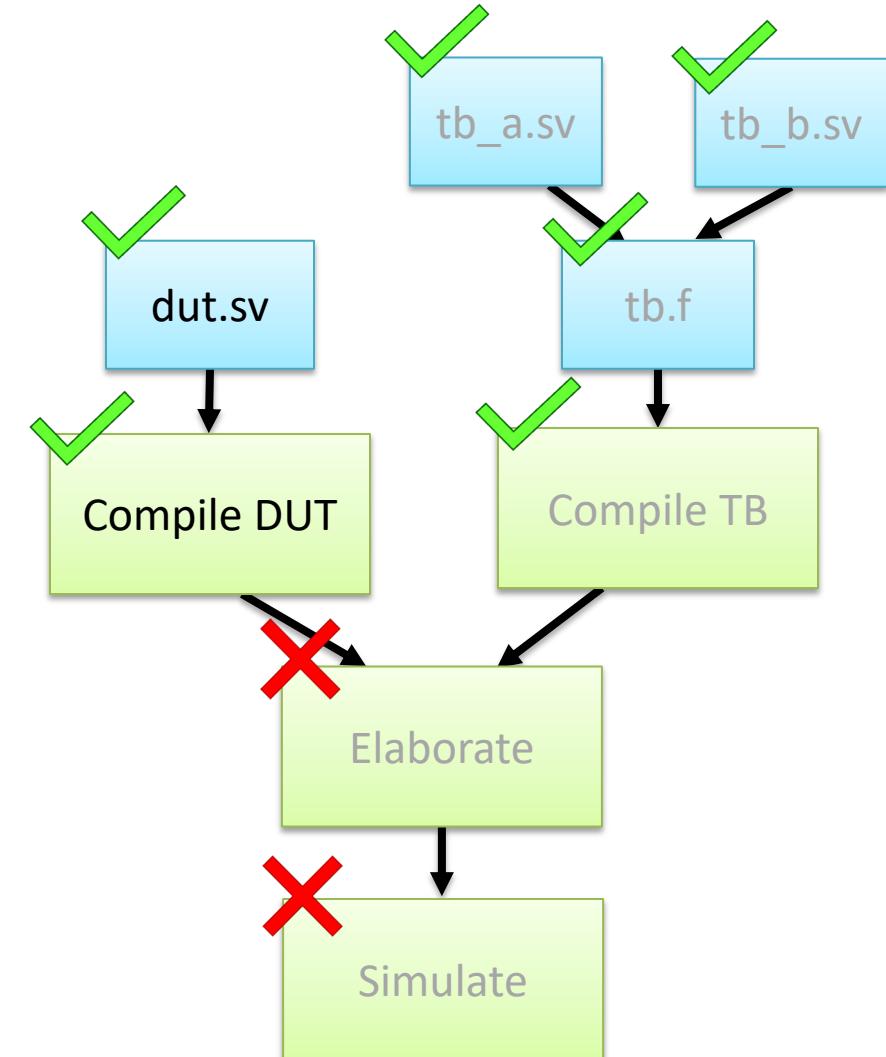
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB
  - Elaborate
  - Start a Simulation
- Next pass: “scons simulate”



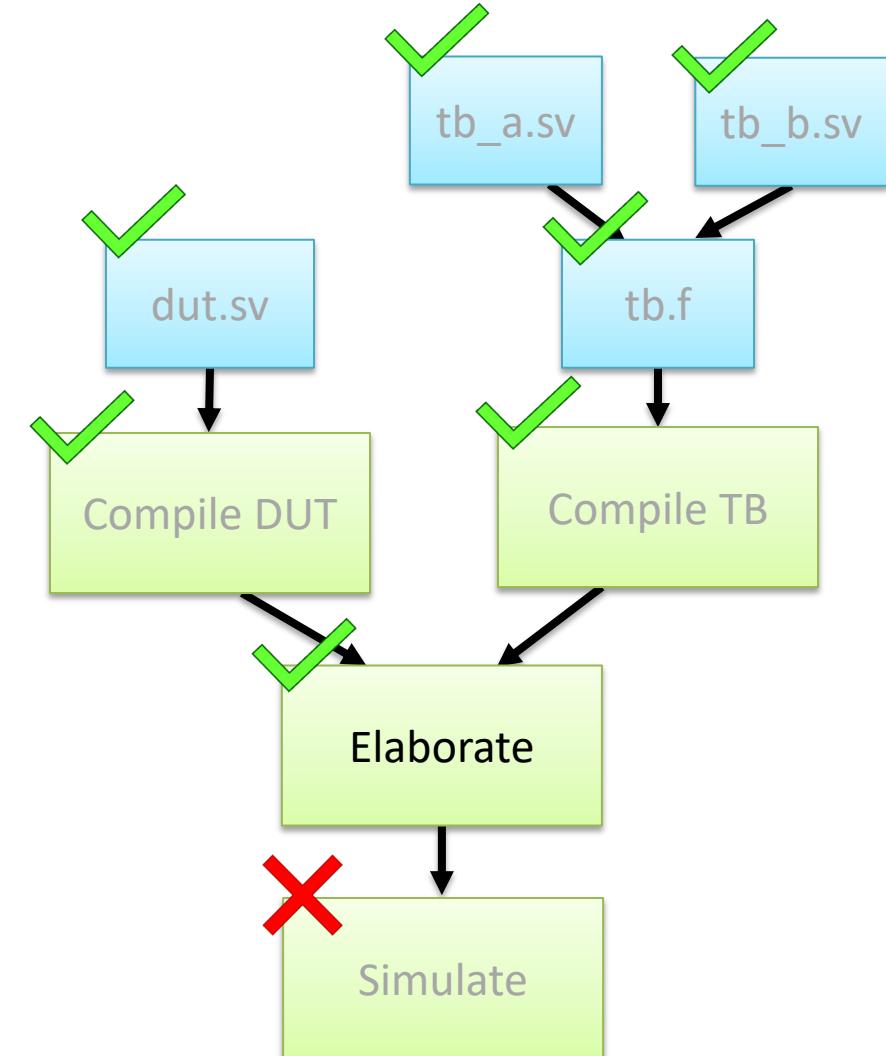
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB
  - Elaborate
  - Start a Simulation
- Next pass: “scons simulate”
  - Only dut.sv was changed, so no need to compile TB



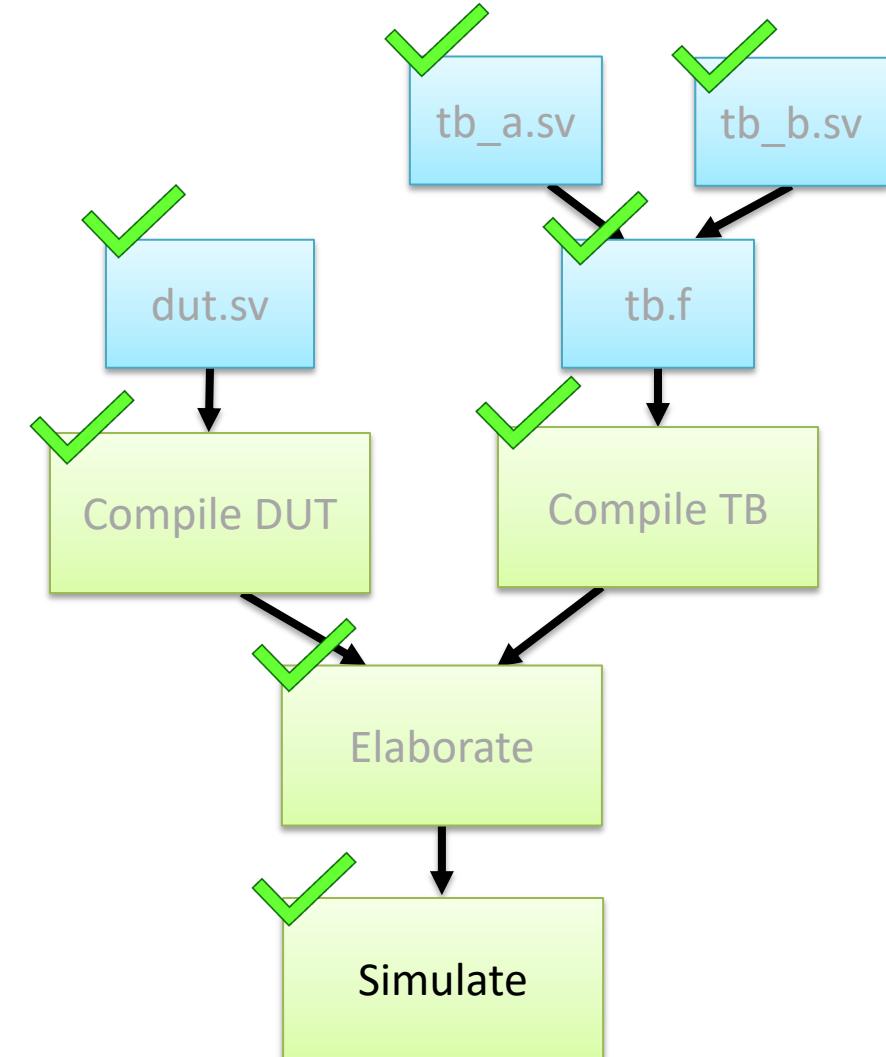
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB
  - Elaborate
  - Start a Simulation
- Next pass: “scons simulate”
  - Only dut.sv was changed, so no need to compile TB



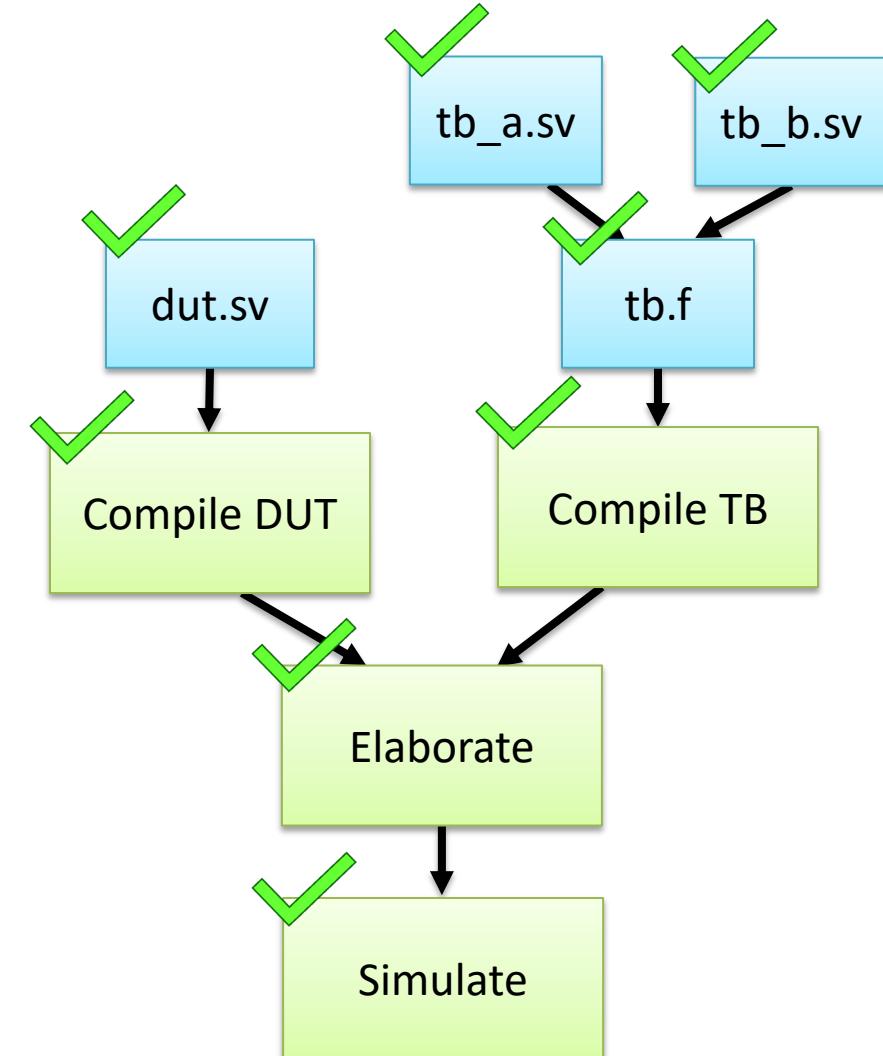
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB
  - Elaborate
  - Start a Simulation
- Next pass: “scons simulate”
  - Only dut.sv was changed, so no need to compile TB



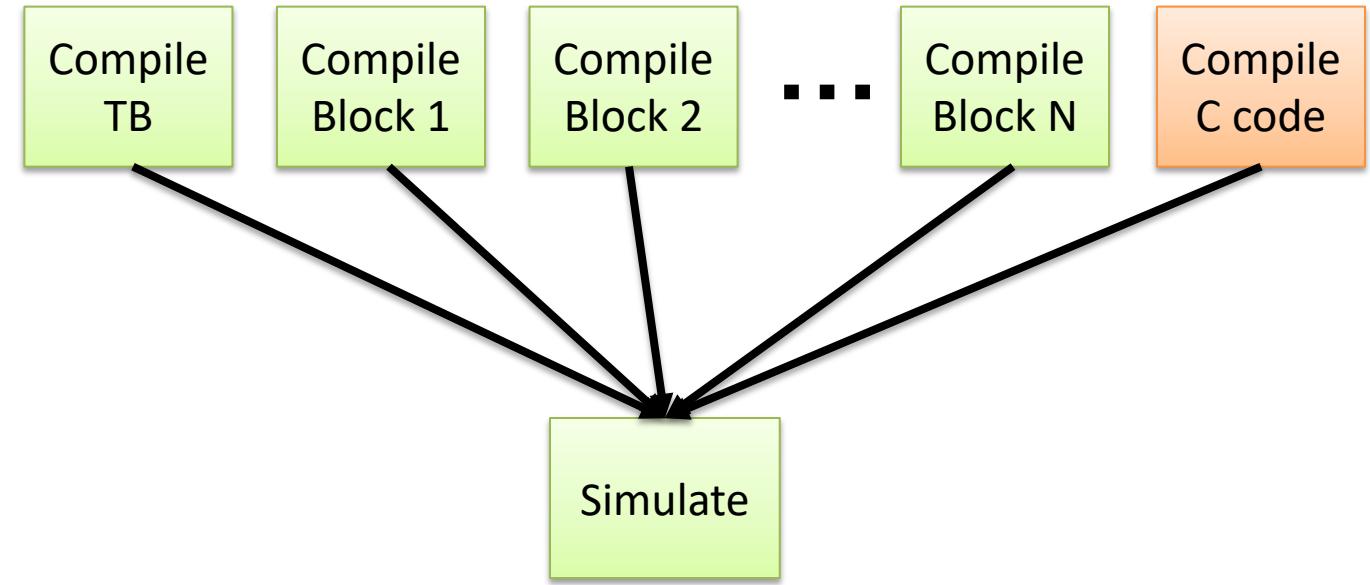
# Automatic Dependency Tracking

- First pass: “scons simulate” at the command line
  - Compile the DUT
  - Compile the TB
  - Elaborate
  - Start a Simulation
- Next pass: “scons simulate”
  - Only dut.sv was changed, so no need to compile TB



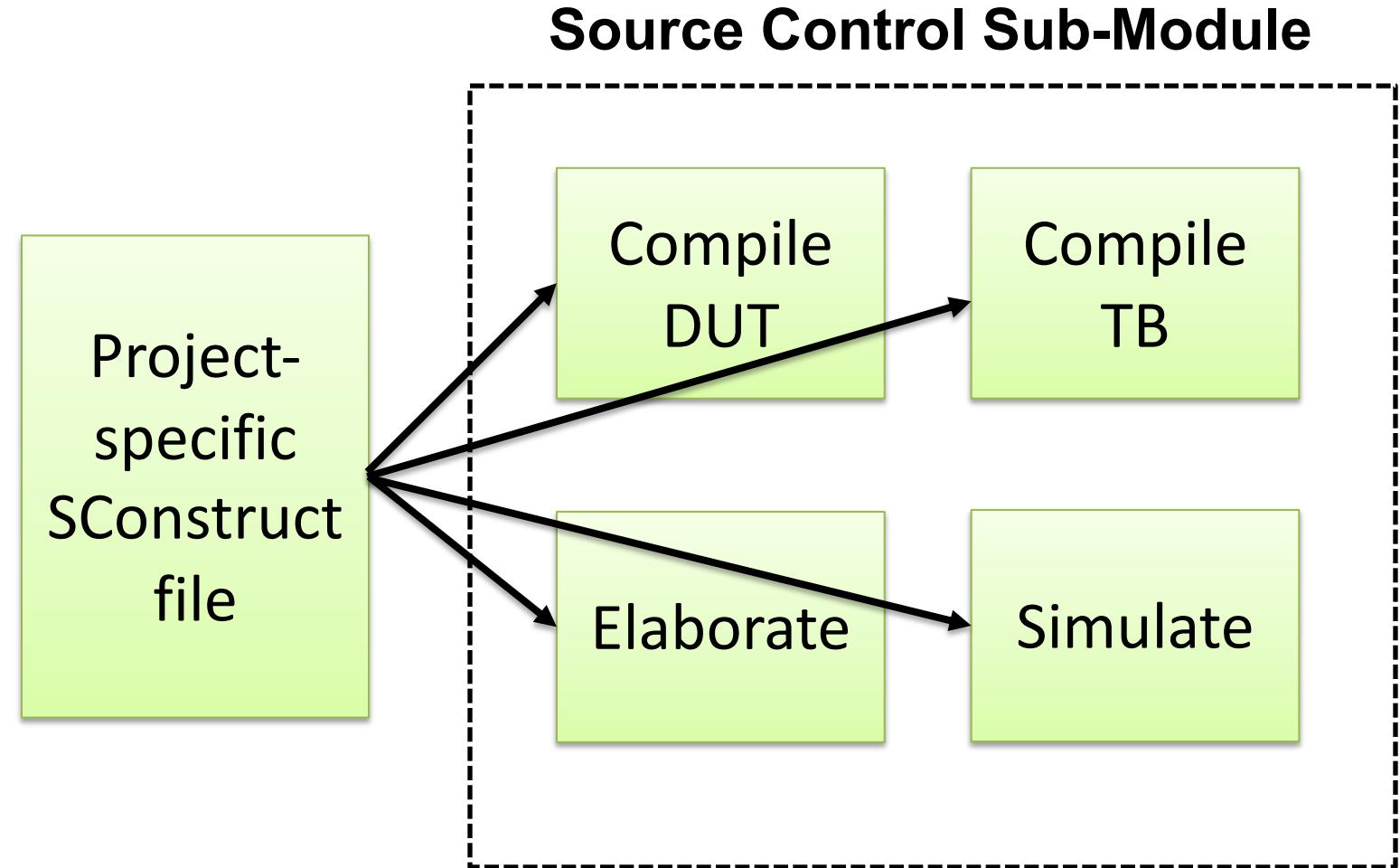
# Automatic Dependency Tracking

- Compile each block, TB, C code in parallel
- Only recompile changed sub-blocks on future passes
- Sub-block compiling can be delegated to the sub-block project



# Centralized Commands

- Keep base commands and default arguments in source control and share it with each project
- Configure project-specific options in the local SConstruct file
- Keeps default commands simple, just a function call



# What Does It Look Like?

## User-facing code

```
env = Environment(...)

com  = env.Com('com.log', 'files.f')
elab = env.Elab('elab.log', com)
sim  = env.Sim('sim.log', elab)
Alias('sim', [sim])
```

## Command-line

```
> scons sim
```

## Centralized code

```
...

def Com(env, target, source, ...):
    return '$COM_CMD -f $SOURCE -l $TARGET'

def Elab(env, target, source, ...):
    return '$ELAB_CMD -lib $WORK -l $TARGET'

def Sim(env, target, source, ...):
    return '$SIM_CMD -lib $WORK -l $TARGET'

...
```

See the paper for an in-depth tutorial and example on using SCons

# Some Additional Features

- Uses a self-contained environment, allowing for consist, portable, and clear build environments
- Cross-platform
- Use Python to programmatically set up
  - simulation targets
  - regressions
  - command line arguments
  - randomize seeds
  - And more!
- Detect file changes using MD5 instead of timestamp
- Create pseudo-builder python functions to do maintenance tasks such as
  - setting up work libraries
  - generating or publishing coverage
  - generating TCL simulation files
  - And more!
- Extensible, add support for any other needed languages or commands

# Conclusion

- SystemVerilog Configurations
  - Now everyone knows how to set up configs for the tools
  - (except those who don't read the paper)
- SCons
  - Like Make but...
  - better - faster - stronger
    - We can build it better and faster, "we have the [better] technology"

