cadence®

Cadence Design Systems, Inc.

RAPID ADOPTION KIT

Static Timing Analysis

Using

TEMPUS

Lab Instruction Document

Note: Testcase Database, scripts and references can be found at the 'Attachments' section below the PDF on Cadence Support website i.e. https://support.cadence.com. This PDF can be searched with the document Title on the portal.

Tool Version
TEMPUS 20.1 (SSV20.10)
September – 2020

Contents

Static Timing Analysis – LABS

I. INTRODUCTION

In the following labs, you will learn how to perform static timing analysis on a Dual Tone Multi Frequency (DTMF) design in each of the four modes and how to run Common User Interface and Unified Metrics using Cadence Tempus

Tempus STA Single Mode Single Corner (SMSC)
Tempus STA Distributed Multi-Mode Multi-Corner (DMMMC)
Tempus STA Concurrent Multi-Mode Multi-Corner (CMMMC)
Tempus STA Common UI (CUI)
Tempus DSTA Block Scope Timing Analysis (STA)
Tempus DSTA Distributed STA on multiple clients Tempus (DSTA)

At the end of the lab sessions, you will be able to:

- Understand how to setup and run each method of the Tempus
- Report and analyze Static Timing results
- Learn how to start a new design and choose the methods that are best for you

---

A. Directory Structure

The lab directory structure is as follows:

<my_dir>/
design/
docs/
libs/
basic_sta/
cmmmc/
dmmmc/
common_UI/
dsta/
unified_metrics/

The docs/ directory contains the Static Timing Analysis Lab Instructions:
Tempus_181_STA_Labs.pdf

You may review the slide presentation on attachments before starting the labs and use them as a reference throughout the labs.

B. Tools and Scripts

This lab uses the Cadence tool named Tempus. Make sure you have the following versions installed.
TEMPUS 17.1 or newer

C. Terminology and Typographical Conventions

Please note the follow key acronyms:
ECO = Engineering Change Order
Tempus = SignOff Static Timing Analysis software
SMSC = Single Mode Single Corner
MMMC = Multi Mode Multi Corner
D-MMMC = Distributed view MMMC timing analysis environment in Tempus
C-MMMC = Concurrent view MMMC timing analysis environment in Tempus
View = Combination of a corner and a timing mode
GTD = Global Timing Debug environment available in the Tempus Graphical Interface
DRV = Design Rule Violation, such as max_cap and max_tran.
Tempus STA = Tempus that runs one job on one machine
Tempus DSTA = Tempus –distributed that runs one job across multiple clients.
LSF = Load Sharing Facility used to access the unix farm of processors

Throughout this lab, anything to be typed will be shown in bold italics, along with a command prompt. Example,

- ***cd $DIR/work/***

Also, during this lab, anything printed in "blue `Courier font style`" is representing information that can be visible in a file (either a script or a log file)

D. Design

The sample testcase is a generic DTMF design, it contains:

- Approximately 10k instances
- Sequential, combinational, and state machine logic
- The following hard macros: PLL, RAM, and ROM.
- Seven clocks
- Up to 8 Views (2 libraries x 2 corners x 2 modes) :
    - Libraries are : Worst and Typical
    - Corners are: RCmax, RCmin
    - Modes are : Functional, Scan

Note: This design is mapped on Cadence open source Libraries, so this explains why the clock period and cell delays may be larger than typically found in a modern design.

E. Lab Flow

The following describes the overall Static Timing Analysis methodology for the labs; this is also described in the **Static Timing Analysis presentation document**:

1. **Run a basic single mode single corner design using Tempus STA**:
   Tempus STA will be used to read in a simple design and generate timing reports. The scripts will highlight the most basic syntax for importing a design and generating timing reports.

2. **Run a concurrent multi-mode multi-corner using Tempus STA with C-MMMC**:
   Tempus STA with the Concurrent MMMC method to time multiple timing views. C-MMMC will generate timing information for multiple views with one Tempus session. The number of views run concurrently is only limited by the amount of available memory.
   There will be one logfile and one combined report generated.

3. **Run a distributed multi-mode multi-corner design using Tempus STA with D-MMMC**:
   Tempus STA with the Distributed MMMC method to time multiple timing views with scripts run in parallel.
   The views will be run in parallel with multiple Tempus sessions.
   D-MMMC will generate timing information per views.

The views are distributed across multiple Tempus sessions. There will be multiple logfiles and one report_analysis_summary file to provide a consolidated cross-view report.

4. **Run Tempus STA with the common UI**:
   Use Tempus STA with an alternate user interface that is common between Cadence Genus and Innovus products. Both single and multiple view setups will be demonstrated.

5. **Run a design using Tempus DSTA**:
   Use Tempus DSTA with Concurrent MMMC method to demonstrate how a large design can be setup and reported. DSTA is the most efficient use of runtime and memory for very large designs. Tempus DSTA is the best solution for design with 50 million of instances or STA runtimes that are well beyond an overnight runtime. Tempus DSTA will use more threads by using more machines in parallel on the same job.
   There will be one main log file, one combined report even though many machines have been used in parallel.

6. **Run a basic single mode single corner design with unified metrics:**
   Tempus STA will be used to read in a simple design and generate timing reports. The scripts will highlight the most basic syntax for creating the snapshot, push and pop operations and report the values in an html file.

Lab 1: Single Mode Single Corner with Tempus STA

F. Lab1.1 : Scripting and logs

Overview: Demonstrate Static Timing Analysis flow using Tempus STA software.

Goals: Learn the most simple STA script. Get familiar with STA reports.

Lab Steps:

1. Inspect the script that loads the design in to Tempus

- ***cd <RAK_DIR>***
- ***cd basic_sta***
- ***cd work***
- **less *../scripts/run.tcl***

This file contains a simple example or reading a design and creating reports for a single set of libraries and timing constraints.
The main commands will look like these:

```
read_lib     ../libs/liberty/FreePDK45_lib_v1.0_worst.lib
read_lib     ../libs/MACRO/LIBERTY/pllclk.lib
read_lib     ../libs/MACRO/LIBERTY/ram_256x16A.lib
read_verilog ../design_hier/dtmf_recvr_core.v.gz
set_top_module dtmf_recvr_core -ignore_undefined_cell

read_spef   ../design/SPEF/corner_worst_CMAX.spef.gz
read_sdc    ../design/dtmf_recvr_core.pr.sdc

# SI delay analysis
set_delay_cal_mode -siAware true
set_si_mode -enable_delay_report true

#glitch analysis
set_si_mode -enable_glitch_report true
set_si_mode -enable_glitch_propagation true

update_timing -full
report_timing
report_noise -txtfile ../scripts/reports/glitch.txt
```

Above flow is unified glitch flow where user is running timing and glitch in single run. There is standalone glitch flow too if user wants to do glitch analysis only.

```
read_lib     ../libs/liberty/FreePDK45_lib_v1.0_worst.lib
read_lib     ../libs/MACRO/LIBERTY/pllclk.lib
read_lib     ../libs/MACRO/LIBERTY/ram_256x16A.lib
```

```
read_verilog ../design_hier/dtmf_recvr_core.v.gz
set_top_module dtmf_recvr_core -ignore_undefined_cell

read_spef    ../design/SPEF/corner_worst_CMAX.spef.gz
read_sdc     ../design/dtmf_recvr_core.pr.sdc

#SI analysis
set_delay_cal_mode -siAware true

#glitch analysis
set_si_mode -enable_glitch_report true
set_si_mode -enable_glitch_propagation true

update_glitch

report_noise -txtfile ../scripts/reports/glitch.txt
```

2. Inspect the script that creates reports from Tempus
   - ***less ../scripts/reports.tcl***

This file contains a small set of reports that describe the design, constraints and timing results.

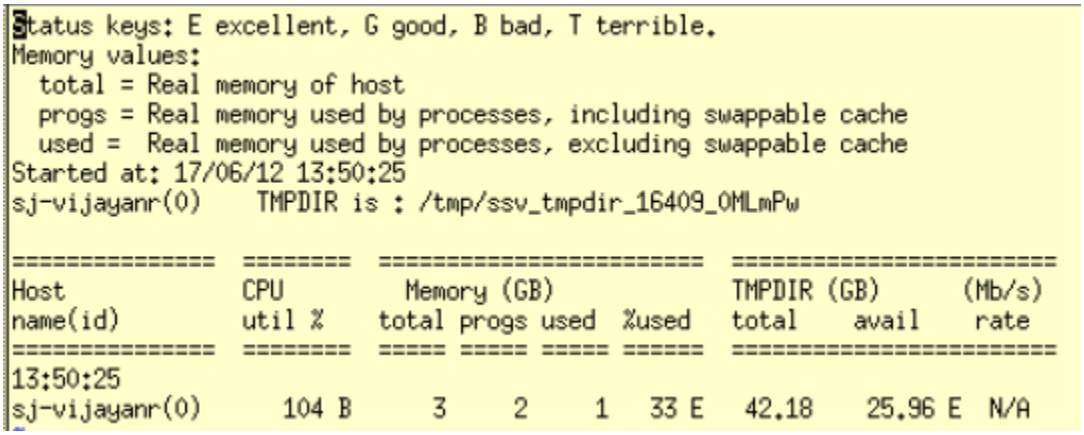
```
check_design -type all -out_file design.rpt
report_annotated_parasitics
report_analysis_coverage

report_clocks
report_case_analysis
report_inactive_arcs

report_constraint -all_violators
report_analysis_summary
report_timing -path_type summary_slack_only -late -max_paths
report_timing -late   -max_paths 1 -nworst 1
report_timing -early  -max_paths 1 -nworst 1
report_timing -retime path_slew_propagation -max_paths 50 -nworst 1
```

3. Start Tempus STA without the GUI
   - ***tempus –nowin***
   - ***monitor_hosts***
   - ***source ../scripts/run.tcl***

   or alternatively you can use
   - ***tempus –nowin –init ../scripts/run.tcl***

This will run TEMPUS to perform Static Timing with SI timing analysis, and create reports. For this tiny design runtime will be just a few seconds. The intent is to show the commands and reports, not to simulate expected runtimes.

Check the ***host-monitor.log*** file to check the CPU usage for running the analysis



```
Status keys: E excellent, G good, B bad, T terrible.
Memory values:
  total = Real memory of host
  progs = Real memory used by processes, including swappable cache
  used =  Real memory used by processes, excluding swappable cache
Started at: 17/06/12 13:50:25
sj-vijayanr(0)    TMPDIR is : /tmp/ssv_tmpdir_16409_OMLmPw

==============  ========  ======================  ======================
Host            CPU         Memory (GB)           TMPDIR (GB)      (Mb/s)
name(id)        util %    total progs used  %used   total    avail    rate
==============  ========  ===== ===== ===== ======  ======================
13:50:25
sj-vijayanr(0)     104 B      3     2     1   33 E   42.18    25.96 E  N/A
```

➢ ***report_resource displays the peak memory used to run the analysis***

```
(total cpu=0:00:11.0, real=0:00:36.0, peak mem=720.4M current
mem=672.8M)
```

From this analysis form, please answer the following questions:

4. Find the command that reports the parasitic annotations.

➢ ***report*anno****

```
ambiguous command name "report*anno*":
report_annotated_assertions
report_annotated_check
report_annotated_delay
report_annotated_parasitics
```

5. Find the available command options by using –help

➢ ***report_annotated_parasitics –help***

```
Usage: report_annotated_parasitics [-help] [-list_annotated] ...

-help                          # Prints out the command usage
-list_annotated                # Generates list of annotated nets
                               # (bool, optional)
-list_float_net                # Generates list of floating nets
                               # (bool, optional)
```

6. Exit tempus

➢ ***exit***

7. Inspect the tempus.log file

The purpose of tempus.log is to capture the run messages and errors.
The filename will increment to tempus.log2 if you run the script again.

➢ ***tail tempus.log***

Notice the runtime and memory message on exit.

```
*** Memory Usage v#3 (Current mem = 672.801M ***
--- Ending "Tempus Timing Signoff Solution" (totcpu=0:00:11.0,
real=0:00:36.0, mem=672.8M) ---
```

8. Inspect the tempus.logv file

The purpose of tempus.logv is to include time stamps for each command.
You can find out when the script started by looking for “license checkout“

```
[02/01 19:51:40     0s] Tempus_Timing_Signoff_XL 18.1 license
checkout succeed.
```

You can find the runtime for each command.
Notice that these commands took about 2 seconds each.

```
[01/17 13:20:48     2s] <CMD> read_lib . . .
[01/17 13:20:48     2s] <CMD> read_verilog . . .
[01/17 13:20:49     2s] <CMD> set_top_module dtmf_recvr_core
```

You can find out when the script finished by looking for “Ending“

```
[02/01 19:52:15    11s] --- Ending "Tempus Timing Signoff Solution"
```

9. Inspect the tempus.cmd file

The purpose of tempus.cmd is to print the list of commands that were run.
When multiple TCL files are source and conditionally branched it might not be clear which commands were run. The tempus.cmd file shows the commands in the order they were executed without comments or multiple files.

10. Inspect the glitch.rpt file

The report_noise command is used to generate text based noise reports. The glitch violations are reported in the glitch.txt file. The file provides insight on the violations and the noise nets.

```
Glitch Violations Summary :
--------------------------
Number of DC tolerance violations (VH + VL) =  1
Number of Receiver Output Peak violations (VH + VL) =  0
Number of total problem noise nets =  1
```

G. Lab1.2 : Reading Design Reports and Timing Reports

1. Inspect reports/annotated.rpt

Was the spef complete?

2. Inspect reports/coverage.rpt

Were the constraints complete?

3. Inspect reports/clocks.rpt
How many clocks were used?

4. Inspect reports/case_analysis.rpt
How many constants were used?

5. Inspect reports/inactive_arcs.rpt
Were a lot of paths impacted by constants?

6. Inspect reports/allviol.rpt
How many setup violations are reported?

7. Inspect reports/analysis_summary.rpt
Is the worst timing do in inputs or reg-reg paths?

8. Inspect reports/start_end_slack.rpt
Is there a single Startpoint that causes the timing issues?

9. Inspect reports/allviol.rpt
- grep "Slack Time" reports/worst_min_path.rpt
- grep "Slack Time" reports/ worst_max_path.rpt

What is the worst setup and hold violation?

Lab 2: C-MMMC with Tempus STA

H. Lab2 : Working with multiple views with C-MMMC

Overview: Demonstrate how to setup Tempus STA to use the C-MMMC method to produce timing report for many views in a single run. All timing views are available in on Tempus session concurrently.
This method describes how to analyze multiple views in one script.

Goal: To demonstrate the definition of timing views. To show how C-MMMC uses a single Tempus sessions to report timing on many views.

Advantages: With one setup file many views can be analyzed.
Using CMMMC your run script analyzed timing for all the views at once. Combined reports were created to capture the timing results for all views. This has a distinct advantage of having one script, one log, one set of licenses, and one set of reports.

Discussion: How to setup a view definition file
A view definition file is the first step. It describes the libraries, constraints and PVT corners to use.

create_library_set: Is used to list the libraries that belong together.
For example the slow libraries would be one group and the fast libraries would be a second group.

create_rc_corner: is used to define the process corners
For example worst_RCMAX or best_RCMIN

create_delay_corner: is used to pair libraries and corners

create_constraint_mode: Is used to define the SDC files for various operating mode.
For example the design may have functional shift and capture operating modes.

create_analysis_view: is used to combine the constraints and delay_corners
For example scan shift may not make much sense outside of typical conditions.
Functional mode may need to be tested with many libraries and RC extractions.

set_analysis_view: is used to turn on the views that you want to run with.
For example you may have 36 unique views but are only interested in timing a few of the most challenging ones today. It is very normal to define all reasonable timing views and work with just the worst ones for now. Signoff will likely use all the views.

Lab Steps:

1. Inspect the view definition file

- ***cd <RAK_DIR>***
- ***cd cmmmc***
- ***cd work***
- ***less ../scripts/viewDefinition_8.tcl***

This small view definition file has been created named viewDefinition_8.tcl .
It has just 8 views to keep the file short for training purposes.

Notice the following sections:
create_library_set
create_rc_corner
create_delay_corner
create_constraint_mode
create_analysis_view
set_analysis_view

2. Inspect the run script

- ***less ../scripts/run.tcl***

Notice the main addition to the script over SMSC is the following line:
read_view_definition ../scripts/viewDefinition_8.tcl

3. Run Tempus C-MMMC script

- ***tempus –nowin***
- ***source ../scripts/run.tcl***

The single log can be found at work/tempus.log

Discussion: save_design

On a large design there can be hundreds of verilog files, library files, dozens of spef files and many thousands of lines of constraints. Loading the design and updating timing information could take a long time. The save_design command saves you time for the next run. If you save the design state it can be restored quickly without having to update_timing again. In a new tempus prompt you then can use restore_design in future weeks and revisit the timing analysis if needed.

Lab 3 : D-MMMC with Tempus STA

I. Lab3.1 : Working with multiple views with D-MMMC

Overview: Demonstrate how to setup Tempus STA to use the D-MMMC method to produce timing report for many views. D-MMMC takes a run script and multiple views and runs each view on a different machine.
This method describes how to distribute multiple views on multiple Tempus STA sessions.

Goal: To demonstrate the definition of timing views. To show how D-MMMC uses multiple Tempus sessions to report one view at a time.

Advantages: As the number of views increase so would the memory and runtime of C-MMMC. D-MMMC enables more views to be run my analyzing views on in separated STA runs on different machines.

Discussion: How to setup a view definition file
The view setup is the same for D-MMMC and C-MMMC.
No changes are required.

How to use a view definition file with D-MMMC
set_distribute_host: Defines how Tempus can grab other machine to clone the job.

distribute_read_design: Define how to load the design and where to put the logs from each Tempus run. One run will be done for each view.

distribute_views: Defines the views to distribute to other Tempus runs and where the top script exists

Lab Steps:
1. Inspect the view definition file
- ***cd <RAK_DIR>***
- ***cd dmmmc***
- ***cd work***
- ***less ../scripts/viewDefinition.tcl***

Notice the following sections are the same as the previous lab:
create_library_set
create_rc_corner
create_delay_corner
create_constraint_mode

create_analysis_view
set_analysis_view

2. Inspect the run script

➢ ***less ../scripts/run.tcl***

```
set_distribute_host -local
# Good for this small RAK design.

distribute_read_design -design_script
../scripts/loadDesign_dmmmc.tcl -outdir $outDir

distribute_views -views [list \
  func_fast_RCMAX \
  scan_slow_RCMIN \
  ] -script ../scripts/sta_dmmmc.tcl
```

3. Run Tempus D-MMMC script

➢ ***tempus –nowin***

➢***source ../scripts/run.tcl***

The script is set to use local CPU. To see distributed clients you can set the `runLocal` variable to 0 and edit the `set_distribute_host` command to appropriate scheduler.

Logs will live under dmmmc_logs

Reports for each view will be created under:
dmmmc_logs/func_slow_RCMIN
dmmmc_logs /scan_fast_RCMAX
dmmmc_logs /func_fast_RCMIN
dmmmc_logs /scan_slow_RCMIN
dmmmc_logs /func_slow_RCMAX
dmmmc_logs /scan_fast_RCMIN
dmmmc_logs /scan_slow_RCMAX
dmmmc_logs /func_fast_RCMAX

Log files will be created with names similar to:
dmmmc_logs /14185_0.log
dmmmc_logs /14185_1.log
dmmmc_logs /14185_2.log
dmmmc_logs /14185_3.log

4. Saving and restoring the design

You can save and restore the design from the tempus terminal.

Save the design using the command

```
distribute_views -save_design all
```

Restore the design using the command

```
distribute_read_design -restore_dir <path>
```

Specify the directory location of saved design views (that was previously saved using `distribute_views -save_design` command)

Note: - In case of DMMMC flow, user cannot execute save_design command after exit_server because there is no design loaded in the master.

To save the design in distributed mode, you need to specify save_design option in distribute_views cmd.

So, execute below command to save Db of all views

```
distribute_views -save_design all
```

If you want to dump Db of selected view, then execute below command

```
distribute_views -save_design list_of_views
```

5. Inspect the tempus.log

To list the views that were analyzed you can search for "Analysis View:" in each log file.

➢ ***grep "Analysis View:" *.log***

```
11557_0.log: Analysis View: func_slow_RCMAX
11557_0.log: Analysis View: func_fast_RCMAX
11557_0.log: Analysis View: scan_fast_RCMIN

11557_1.log: Analysis View: func_slow_RCMAX
11557_1.log: Analysis View: func_slow_RCMAX
11557_1.log: Analysis View: scan_slow_RCMIN

11557_2.log: Analysis View: func_slow_RCMAX
11557_2.log: Analysis View: func_fast_RCMIN
11557_2.log: Analysis View: scan_slow_RCMAX

11557_3.log: Analysis View: func_slow_RCMAX
11557_3.log: Analysis View: func_slow_RCMIN
11557_3.log: Analysis View: scan_fast_RCMAX
```

The views being analyzed will be described in the log file like this:

```
Analysis View: func_slow_RCMAX
    RC-Corner Name        : corner_worst_RCMAX
    RC-Corner Index       : 0
    RC-Corner Temperature : 25 Celsius
```

```
RC-Corner Cap Table   : ''
RC-Corner PreRoute Res Factor         : 1.2
RC-Corner PreRoute Cap Factor         : 1.2
RC-Corner PostRoute Res Factor        : 1 {1.2 1.2 1.2}
RC-Corner PostRoute Cap Factor        : 1 {1.2 1.2 1.2}
RC-Corner PostRoute XCap Factor       : 1 {1.2 1.2 1.2}
RC-Corner PreRoute Clock Res Factor   : 1.2
RC-Corner PreRoute Clock Cap Factor   : 1.2
RC-Corner PostRoute Clock Cap Factor  : 1 {1.2 1.2 1.2}
RC-Corner PostRoute Clock Res Factor  : 1 {1.2 1.2 1.2}
```

Comments:
Using DMMMC your run script was multiplied and applied to each view.
This was very much like running 8 different SMSC runs and getting 8 directories of results.

Since this design was small, 8 machines were not required by the set_distribute_host command. Typically 8 machines and 8 Tempus sessions would have been used to maximize throughput and a large design.

D-MMMC is typically used when C-MMMC runtime needs to be reduced or memory is limited. D-MMMC works well when dozens or hundreds of views are required. If there were only 8 views it is likely that C-MMMC would have been a fine choice since this designs does not consume much runtime or memory.

Note :-If user wants to debug the issue of any particular view, then he/she needs to execute that cmd ..

set_distribute_mmmc_mode -interactive true -views all

With this command, user can check timing of any specific view through master.
To exit from interactive mode, please switch the argument to false.

set_distribute_mmmc_mode -interactive false

For interactive mode, user must ensure number of views equal to the number of remotehost.

Lab 4: Tempus DSTA for large designs

J. Lab4.1 : Working with large design with Tempus DSTA

Overview: Demonstrate how to setup Tempus DSTA. This is a separate product beyond Tempus STA that enables much higher design capacities. DSTA takes your design and partitions it across multiple client machines. In this way the workload and memory is shared. Designs that are too large to fit on one machine can now use DSTA to go to the next level of capacity.
This method is not related to distributing views. DSTA is distributing any design across multiple machines for maximum capacity and utilizes average computer hardware.

Goal: To demonstrate the setup of Tempus DSTA. To show how multiple clients can work together to perform similar work as the C-MMMC lab. This lab uses 8 views and C-MMMC.

Advantages: Many machines with 128Gig of ram can be used rather than needing a 1024 Gig on a single machine. Designs with 400 million instance are now possible without any design abstractions. This has a distinct advantage of having one script, one log, one set of licenses, and one set of reports.

Discussion of STA vs DSTA: The main difference between Tempus STA and DSTA is unlocking the power of multiple machines. Tempus DSTA has a master process and client machines that do most of the work. Runtime and memory is decreased each time you add more clients.

DSTA uses a master process with multiple threads.
The master dispatches work to multiple clients that are also multi-threaded.
The number of clients you should use is directly related to design size.

Design Instance Count	Client Cnt	Client Thread Cnt	Master Thread Cnt	Resulting Proc Cnt	Comment
A few million	2	2	2	6	Education purposes only. Good for Tempus RAK training
20 million	5	4	4	24	Small design
50 million	4	8	8	40	Medium design
100 million	8	8	8	72	Larger design
400 million	12	12	12	156	Very large design

Discussion about DSTA scripts: The DSTA scripts are intended to be very similar to all the previous labs. There are a few new areas that tell DSTA how to access more machines and there may be a few syntax additions to familiar commands.

Lab Steps:

1. Inspect the run script

- ***cd <RAK_DIR>***
- ***cd dsta***
- ***cd work***
- ***less ../scripts_smsc/run.tcl***

Notice the following sections unique to DSTA:

```
# There are two new switches for client threading and client count.
set_multi_cpu_usage -localCpu 1 -cpuPerRemoteHost 1  -remoteHost 3

set_distribute_host
distribute_start_clients
read_lib
read_verilog
set_top_module
read_spef
distribute_partition
```

Discussion about thread and client counts : This DSTA scripts will be slightly different for every company and every design. Each design will have its own demands for thread count and client count.
If your company has a lot of small memory machine you may increase your client count to conserve memory. If your company needs more speed you may increase your client count to conserve runtime.
One large 32 proc machine can host 4 clients with 8 threads each. Each client does not have to be a unique machine.
If your design is small you might start with 4 threads and 2-4 clients.
In this lab the design is way below the 20 million instances where DSTA starts to shine.

Discussion about compute infrastructure: This DSTA scripts will be slightly different for every company due to compute infrastructure. Since DSTA needs to spawn jobs to other machine you will need to specify how to get access or more machines. The typical options are LSF, SGE, SSH, RSH. The names of the queues or machines will need to be provided to DSTA.

2. Modify the DSTA settings for you current design.

This lab will not function exactly as written because it needs to be told about your compute environment. For this lab it is assumed that LSF is being used.
Other options for requesting machines are Sun Grid Engine, RSH, or SSH.
Change the LSF queue to match want you have access to.

Change the threadCount to match what is reasonable for your design.
In this academic lab 1-2 threads is more than enough. Typically 4, 8, 12 or 16 will be used.
Change the clientCount to match what is reasonable for your design.
In this academic lab 2 clients is enough. Typically 4, 8, 12 will be used.

Change the LSF queue name to match the queue you have access to.
Does your LSF system demand more syntax? Add it after the set_distribute_host –R option.

Change the memory requirement for each machine. This prevents LSF from sending your large design to a small memory machine. In this case just 8 gig of ram is more than enough. Typically your design will require 64, 128, or 512 gig of ram.

Change the /tmp disk space requirement for each machine. This prevents LSF from sending your large design to a machine with limited /tmp disk space. Each client needs /tmp that is about the size of your spef file. In this case just 1 gig of ram is more than enough. Typically your design will require 10, 50, 100 gig of /tmp scratch space.

To make setup easy the following variables are in the script:

```
set clientCnt           2
set clientThreadCnt     1
set masterThreadCnt     1
set lsfQueue            ssv
set maxMem              8000
set tmpDisk             1000
```

3. Run Tempus DSTA script

- ***tempus -nowin -distributed***
- ***source ../scripts_cmmmc/run.tcl***

Discussion about Tempus DSTA logfiles:
The master process is the one the gives you the command prompt.
It is responsible for reading the design and sends work to the clients.
The master logfile can be found at:
work_dsta/tempus.log

The client process each work on a portion of the design.
Each of the client logs looks like a normal STA log.
The client logs can be found at:
work_dsta/partOutput_0/tempus.log
work_dsta/partOutput_1/tempus.log
work_dsta/partOutput_2/tempus.log

Discussion on the timing reports:
By default all of the timing views are reported in the same reports.
The header named “Analysis View” keeps the timing paths unique for each path.

```
Path: VIOLATED Setup Check with Pin reg[12]/CK
Endpoint: reg[12]/D (^) checked with trailing edge of 'm_rcc_clk'
Beginpoint:  ir_reg[0]/Q (^) triggered by  leading edge of 'm_clk'
Path Groups: {m_rcc_clk}
Analysis View: scan_slow_RCMAX
Other End Arrival Time         21.389
- Setup                         0.805
+ Phase Shift                   0.000
- Uncertainty                   0.400
= Required Time                20.183
- Arrival Time                 20.240
= Slack Time                   -0.057
```

The reports created by the master should look very familiar. This multi-view design creates on report that merges all the results. DSTA give you one place to look for all your results. It you want to report one timing view at a time that is available but the default is to report all enabled timing views.

Comments on the DSTA lab:
This lab is not designed to show off runtime and memory improvements versus increasing client counts. The design is far too small to see any changes. This lab is designed to show the syntax and order of operations.

Using Tempus DSTA on design smaller than 20 million instances will not show any significant gains. Tempus DSTA is designed for large design where Tempus STA with maximum threading is still not enough speed. DSTA is also used where there is not enough memory on any one machine to hold your full design. When Tempus STA is hitting a limit, then Tempus DSTA will be there to cut runtime and memory.

Lab 5: Single Mode Single Corner - Tempus common UI based run in Tempus STA

K. Lab 5.1 : Scripting and logs

Overview: Demonstrate Static Timing Analysis flow in Common UI based run using Tempus STA software.

Goals: Learn Common user interface by analyzing a simple STA script

Lab Steps:

1. Inspect the script that loads the design in to Tempus

- ***cd <RAK_DIR>***
- ***cd common_UI***
- ***cd work***
- **less *../scripts/run.tcl***

This file contains a simple example or reading a design and creating reports for a single set of libraries and timing constraints.
The main commands will look like these:

```
set_db timing_analysis_type ocv
set_db timing_analysis_cppr both

#SI delay analysis
set_db delaycal_enable_si true
set_db si_delay_separate_on_data true
set_db si_delay_enable_double_clocking_check true
set_db si_delay_enable_report true

#glitch analysis
set_db si_glitch_enable_report true
set_db si_glitch_enable_propagation true

update_timing -full
report_timing -to out -fields {instance cell arc transition load
delay incr_delay }

report_noise -txtfile ../scripts/reports/glitch.txt
```

2. Inspect the script that creates reports from Tempus

- ***less ../scripts/reports.tcl***

This file contains a small set of reports that describe the design, constraints and timing results.

```
check_design -type all -out_file design.rpt
```

```
report_annotated_parasitics
report_analysis_coverage

report_clocks
report_case_analysis
report_inactive_arcs

report_constraint -all_violators
report_analysis_summary
report_timing -late   -max_paths 1 -nworst 1
report_timing -early  -max_paths 1 -nworst 1
```

3. Start Tempus STA without the GUI

- ***tempus –nowin –common_ui {tempus -nowin -stylus*** ➔ **Tempus 19.1 release and onwards*}***
- ***monitor_distributed_hosts***
- ***source ../scripts/run.tcl***

or alternatively you can use

- ***tempus –nowin –common_ui –init ../scripts/run.tcl***

This will run TEMPUS to perform Static Timing with SI timing analysis, and create reports. For this tiny design runtime will be just a few seconds. The intent is to show the commands and reports, not to simulate expected runtimes.

- ***report_resource displays the peak memory used to run the analysis***

```
Current (total cpu=0:00:18.4, real=0:00:30.0, peak res=780.5M,
current mem=686.7M)
```

6. Exit tempus

- ***exit***

Comments:

The Stylus Common User Interface (Common UI from this point onwards) has been designed to be used across Genus, Joules, Modus, Innovus, Tempus, and Voltus tools. By providing a common interface from RTL to signoff, the Common UI enhances user experience by making it easier to work with multiple Cadence products.

The Common UI simplifies command naming and aligns common implementation methods across Cadence digital and signoff tools. For example, the processes of design initialization, database access, command consistency, and metric collection have all been streamlined and simplified. In addition, updated and shared methods have been added to run, define, and deploy reference flows. These updated interfaces and reference flows increase productivity by delivering a familiar interface across core implementation and signoff products. You can take advantage of

consistently robust RTL-to-signoff reporting and management, as well as a customizable environment

Lab 6: Single Mode Single Corner – Unified Metrics in Tempus STA

L. Lab 6.1 : Scripting and logs

Overview: Demonstrate Static Timing Analysis flow with unified metrics using Tempus STA software.

Goals: Learn unified metrics by analyzing a simple STA script

Lab Steps:

1. Inspect the script that loads the design in to Tempus
   - ***cd <RAK_DIR>***
   - ***cd unified_metrics***
   - ***cd work***
   - **less** ***../scripts/run.tcl***

This file contains a simple example or reading a design and creating reports for a single set of libraries and timing constraints.
The main commands will look like these:

```
um::enable_metrics -on
um::push_snapshot_stack

um::push_snapshot_stack

um::create_snapshot –name basic_sta
```

2. Inspect the script that creates reports from Tempus
   - ***less ../scripts/reports.tcl***

This file contains a small set of reports that describe the design, constraints and timing results.

```
check_design -type all -out_file design.rpt
report_annotated_parasitics
report_analysis_coverage

report_clocks
report_case_analysis
```

```
report_inactive_arcs

report_constraint -all_violators
report_analysis_summary
report_timing -late   -max_paths 1 -nworst 1
report_timing -early  -max_paths 1 -nworst 1

um::report_metric -format html -file  ${reportDir}/metrics.html
```

3. Start Tempus STA without the GUI

- ***tempus –nowin***
- ***monitor_hosts***
- ***source ../scripts/run.tcl***

or alternatively you can use

- ***tempus –nowin –init ../scripts/run.tcl***

This will run TEMPUS to perform Static Timing with SI timing analysis, and create reports. For this tiny design runtime will be just a few seconds. The intent is to show the commands and reports, not to simulate expected runtimes.

- ***report_resource displays the peak memory used to run the analysis***

```
Current (total cpu=0:00:18.4, real=0:00:30.0, peak res=780.5M,
current mem=686.7M)
```

6. Exit tempus

- ***exit***

7. Inspect metrics.html file

- Open the metrics.html file
- Unified metrics contains consolidated reports of all the snapshot created during a flow
- Since we have created a single snapshot **basic_sta**, you can see the details regarding the flow in the metrics.html report
- Select **flow** tab to see the total run time and the memory it has taken to run the flow

Comments:

Unified Metrics (UM) is a system integrated with Cadence tools that automatically delivers data about the design and run to you. UM provides functionality to:

- Stream data from the reports and algorithms.
- Collect and organize the stream data into a structured form.
- Display the data in a Simple Unified Metrics file or the Advanced Unified Metrics server.

Tempus Training Next Steps

M. Download labs and rerun at your company

Testcase Database, scripts and references can be found at 'Attachments' section below the PDF.

This PDF can be searched with the document Title on https://support.cadence.com

N. Modify the labs to read in your libraries and designs

Runtime

Snapshot	Runtime				Memory		
	CPU (s)	Total CPU (s)	Wall (s)	Total Wall (s)	Resi+Virtual (MB)	Resident (MB)	Peak Resident (MB)
basic_sta	0:00:21	0:00:21	0:00:24	0:00:24	771	643	643