Cadence Design Systems, Inc.

RAPID ADOPTION KIT

Static Timing Analysis

Using

Block Scope

Lab Instruction Document

Tool Version
TEMPUS 17.1

August – 2017

Note: *Testcase database, scripts and references can be found at 'Attachments' and 'Related Solutions' sections below the PDF on Cadence Support i.e.* https://support.cadence.com*, This pdf can be searched ONLINE with the document 'Title' as well.*

Table of Contents

Static Timing Analysis – LABS

I. INTRODUCTION

In the following labs, you will learn how to perform Tempus timing analysis on your largest chips. Block Scope is the set of features that allows the timer to be focused on the blocks that are changing to improve runtime and capacity. This feature is intended for Tempus DSTA customers that work with hierarchical designs and ECO calculation. It is expected that you have some familiarity with Tempus DSTA. These labs highlight the Block Scope use models.

1. Tempus DSTA Full Chip with Block Scope setup
2. Tempus DSTA Block Scope

At the end of the session, you will be able to:

- Understand how to setup and run each of the Tempus DSTA use models.
- Report and analyze Static Timing results for a Full design.
- Report and analyze Static Timing results for selected Blocks.
- Be confident how to start your new design and choose the methods that are best for you.

A. Directory Structure

The lab directory structure is as follows:

```
<my_dir>/
        libs/
        design/
        docs/
        block_scope/
```

The docs/ directory contain the Static Timing Analysis Lab Instructions and the training slides:

Tempus_171_BlockScope_Labs.pdf
Tempus_171_BlockScope_Slides.pdf

Review the slide presentation before starting the labs, and use them as a reference throughout the labs. The labs and slides complement each other.

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	= Static Timing Analysis software
SMSC	= Single Mode Single Corner
MMMC	= Multi Mode Multi Corner

General terms:

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
Tempus STA	= Tempus that runs one job on one machine
Tempus DSTA	= Tempus –distributed that runs one job across multiple clients.
Tempus TSO	= Tempus Timing Signoff Optimization for ECO calculation
LSF	= Load Sharing Facility used to access the unix farm of processors

Block Scope terms:

Full Chip Analysis	= Traditional timing analysis of the complete design
Block Scope Analysis	= Timing analysis limited to one or more modules
Top Scope Analysis	= Timing analysis of the top design logic with most or all of the sub-modules removed
Timing Context	= The logic and timing environment surrounding a block
Block_list	= A list of modules included in the timer update
Exclude_list	= A list of modules excluded from the timer update

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

➢ ***cd <RAK_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 testcase is a small DTMF design that 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 the clock periods and cell delays may be larger than typically found in a modern design. The design is also very small to ensure fast lab runtimes. This small design is mainly used to highlight command syntax and ordering. The scripts are written to run on a single machine even though Tempus DSTA can run with many clients.

E. Lab Flow

The following describes the overall Static Timing Analysis methodology for the labs.

1. **Run a Full Chip CMMMC script using Tempus DSTA**:
   Tempus DSTA Full Chip script will be used to demonstrate the syntax and order of commands to enable the Block Scope flow. This script is very similar to the traditional DSTA script with a few lines added to enable block scope. This top level run will extract the context that surrounds our selected blocks.

2. **Run a Block Scope CMMMC script using Tempus STA**:
   This lab shows how to analyze a block with the full chip context with a fraction of the resources. This is typically done after each iteration of an ECO to insure the most accurate timing without needed a Full Chip run.

Lab 1: Full Chip Multi-mode Multi-corner with Tempus DSTA

F. *Lab1.1 : Scripting and logs*

Overview: Demonstrate a Full Chip Static Timing Analysis flow using Tempus DSTA.

Goals: Learn how write a very simple DSTA script that enables the use of Block Scope at later time.

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 related to design size.

Steps:

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

The run_full_chip.tcl file contains a simple example of reading a design and creating scopes to enable block scope. This represents a typical Full Chip script with two small sections added to enable Block Scope at a later time.

These sections enable Block Scope:

```
###########################################
# Full Scope step 1
###########################################
set full_context_flow 1 ;# Prepare to save data for block scope

. . .

##################################
# Full Scope step 2
# Write out timing scope directories
##################################
set scopeDir scope_block1
set instanceList TDSP_CORE_INST/ALU_32_INST/sub_84_22
distribute_create_scope -instance $instanceList -path $scopeDir
set scopeDir scope_top
set topDesign [get_property [current_design] hierarchical_name]
Puts "Working on the top design named $topDesign"
distribute_create_scope -cell_nohier $topDesign -path $scopeDir
```

2. Start Tempus DSTA without the GUI
   ➢ ***tempus -distributed***
   ➢***source ../scripts/run_full_chip.tcl***
   alternatively you can use:
   ➢ ***tempus –distributed –init ../scripts/run_full_chip.tcl***

If this top level script passes all signoff timing reports, we would be done.
Since we enabled the option of Block Scope, we can now analyze timing for future block changes without needing another full chip run. Future analysis can use Tempus STA instead of this Full Flat Tempus DSTA script.

4. Exit tempus
➢ ***exit***

5. Inspect the files in your directory

tempus.log	Is your log file.
partOutput_0	Is the work directory for the first client and include its log.
monitor_host.log	Tells you about the machine resources available.
scope_block1	This directory enables block scope on block1
scope_top	This directory enable block scope on the top level design

Discussion about lab1:

The purpose of this lab was to highlight the two small edits required to change any Full Chip script to enable Block Scope. The saved scope directory were created to capture the timing context for the selected blocks. This Full Chip run has now captured all the information to speed up future runs by using Block Scope. If your blocks change you can re-analyze them with full chip accuracy and full chip constraints without requiring Tempus DSTA or a Full Chip run.

Lab 2: Block Scope Analysis with Tempus STA

G. Lab2.1 : Block Scope with Tempus STA

Overview: The block scope script will have very similar timing results to the full chip script, but will use less processor resources. Block Scope will select the modules that update_timing will work on and the surrounding netlist context will be re-analyzed to provide results like the full chip. By focusing resources only on the designs that have changed, runtime and machine resources can be reduced.
A script that only runs timing for a stand-alone block would not have to rely on budgetary timing constraints with some guesses about the drives, loads, and crosstalk effects. Block Scope removes those limitations and times the blocks with the Full Chip constraints, loads, and crosstalk effects.

Goal: To demonstrate the use of Block Scope using Tempus STA.

Advantages: Timing reports use the Full Chip constraints. Timing reports show the full paths of the blocks just like a Full Chip run would show. Block Scope runs may require about half the number of CPUs and still have half the runtime of a Full Chip run. This is possible because your block is likely 25% of the total instance count.

Steps:

1. <u>Inspect the run script</u>

➢ ***cd <RAK_DIR>***
➢ ***cd block_scope***
➢ ***cd work***
➢ ***less ../scripts/run_block1_scope.tcl***

Notice the following sections unique to Block Scope:

====================
Block scope step 1a
====================
```
read_scope_verilog <filenames>
```

====================
Block scope step 1b
====================
```
read_scope_spef <filenames> [-rc_corner <corner> | -early_rc_corner
<corner> | -late_rc_corner <corner>]
set scopeDir scope_block1
read_scope -path $scopeDir
```

2. <u>Run Tempus STA script</u>

➢ ***tempus -nowin***
➢***source ../scripts/run_block1_scope.tcl***

Discussion about Block Scope script:
The read_scope_verilog command would be used to read in any new Verilog you have from an ECO. This step is not required to run Block Scope analysis, but a typical flow would be used to test an updated netlist. Use read_scope_verilog when you have an update netlist to test against with Block Scope.

The read_scope_spef command would be used to read in any new SPEF you have from an ECO. This step is not required to run Block Scope analysis, but a typical flow would be used to test an updated design.

The read_scope command imports the saved scope directory from your full chip run. This directory contains the surrounding context netlist, foreign aggressors and top level timer settings and top level constraints. It is everything you need to test your block with the surrounding logic that make it behave much like a full chip run.

Discussion on the timing reports:
The timing reports will contain Top level gates propagating into your selected module and all top level gates driven by your selected block. Think of this like a full chip design that keeps all paths to, for and inside your selected block. All additional logic has been removed
This is possible because Block Scope includes the fanin and fanout cones of your selected modules. The Block's timing path is listed in its full context that may include some top level gates on the block inputs or outputs.
This report used the ALU module. The instance name in this report is ALU_32_INST.
All the flight time through 19.65 ns is show with full gate details.

```
###############################################################
#  Generated by:      Cadence Tempus 17.90-d053_1
#  OS:                Linux x86_64(Host ID sjfnl155)
#  Generated on:      Mon Jul 17 09:59:52 2017
#  Design:            dtmf_recvr_core
#  Command:           report_timing
###############################################################
Path 1: MET Setup Check with Pin TDSP_CORE_INST/EXECUTE_INST/acc_reg[24]/CK
Endpoint:   TDSP_CORE_INST/EXECUTE_INST/acc_reg[24]/D (^) checked with  leading edge
of 'm
_clk'
Beginpoint: TDSP_CORE_INST/DECODE_INST/ir_reg[11]/Q   (v) triggered by  leading edge
of 'm
_clk'
Path Groups: {m_clk}
Analysis View: scan_slow_RCMAX
Other End Arrival Time          5.228
- Setup                         0.561
+ Phase Shift                  40.000
- Uncertainty                   0.400
= Required Time                44.266
- Arrival Time                 36.698
= Slack Time                    7.568
     Clock Rise Edge                 0.000
```

```
      + Clock Network Latency (Prop)  5.249
      = Beginpoint Arrival Time       5.249
       ---------------------------------------------------------------------------------
----
-------------------------------------------
       Pin                                                                    Edge  Cell
Sl
ew   Load   Delay  Incr   Arrival  Required

                   Delay  Time     Time
       ---------------------------------------------------------------------------------
----
-------------------------------------------
       TDSP_CORE_INST/DECODE_INST/ir_reg[11]/CK                               ^     -
0.
139  0.005  -      -      5.249    12.817
       TDSP_CORE_INST/DECODE_INST/ir_reg[11]/Q                                v     SDFFS_X2
0.
126  0.001  0.651  0.000  5.900    13.469
       TDSP_CORE_INST/DECODE_INST/FE_OFC192_ir_11_/Z                          v     BUF_X16
0.
383  0.059  0.466  0.000  6.366    13.935
       TDSP_CORE_INST/TDSP_CORE_GLUE_INST/lsh_120_47/Fn0145D9125/ZN           ^     INV_X4
3.
444  0.028  2.326  0.000  8.692    16.261
       TDSP_CORE_INST/TDSP_CORE_GLUE_INST/lsh_120_47/p0088D/ZN                ^     AND2_X2
0.
747  0.006  1.145  0.000  9.838    17.406
       TDSP_CORE_INST/TDSP_CORE_GLUE_INST/lsh_120_47/g9132/ZN                 ^     AND2_X2
0.
151  0.001  0.444  0.000  10.282   17.850
       TDSP_CORE_INST/TDSP_CORE_GLUE_INST/lsh_120_47/FE_PSC502_n_661/Z        ^     BUF_X32
0.
094  0.003  0.233  0.000  10.515   18.083
       TDSP_CORE_INST/TDSP_CORE_GLUE_INST/p0171A/ZN                           v     NAND2_X4
0.
560  0.003  0.123  0.000  10.637   18.206
       TDSP_CORE_INST/TDSP_CORE_GLUE_INST/p0089D/ZN                           v     AND3_X2
0.
092  0.001  0.591  0.000  11.229   18.797
       TDSP_CORE_INST/TDSP_CORE_GLUE_INST/p0053D/ZN                           ^     NAND2_X4
1.
440  0.021  0.994  0.000  12.223   19.791
       TDSP_CORE_INST/ALU_32_INST/add_81_22/p1334A/ZN                         v     NAND2_X1
0.
623  0.003  0.621  0.000  12.844   20.412
       TDSP_CORE_INST/ALU_32_INST/add_81_22/p1331A/ZN                         ^     NOR2_X4
0.
481  0.001  0.516  0.000  13.360   20.928
       TDSP_CORE_INST/ALU_32_INST/add_81_22/FE_RC_539_0/ZN                    v     INV_X4
…….
```

H. Lab2.1 : Top Scope with Tempus STA

Overview: Top scope is the same idea as block scope. The only difference is that the Top Level design is the block being selected. This is good for top level timing closure by removing most of the sub_design logic. Top scope is simply Block Scope that has selected the Top design. It is not a separate tool.

If your top level logic and routing is changing, Top Scope will save you from running full chip analysis each time. Top Scope analysis will let you evaluate top level logic and routing changes efficiently.

Goal: To demonstrate the use of Top Scope using Tempus STA.

Advantages: Analyze top level logic and routing changes without evaluating all the sub-designs internal paths.

Steps:

3. Inspect the run script

➢ ***cd <RAK_DIR>***
➢ ***cd block_scope***
➢ ***cd work***
➢ ***less ../scripts/run_top_scope.tcl***

Notice the following sections unique to Block Scope:

```
read_scope_verilog <filenames>
read_scope_spef <filenames> [-rc_corner <corner>
set scopeDir scope_block1
read_scope -path $scopeDir
```

4. Run Tempus STA script

➢ ***tempus -nowin***
➢***source ../scripts/run_top_scope.tcl***

Discussion about Top Scope script:
Notice that the Top Scope script has the same syntax as block scope.
This script is provided to show the flow.

The timing reports will contain Top level gates, block input-> register and block register -> out gates will be analyzed. The block register-> register paths will not be analyzed or reported. All the gate that interact with the top level design is what the timer update with focus on. This is similar to and ILM flow.

Comments on the Tempus lab size:
The design is far too small to see any Full Chip vs Block Scope runtime changes.

I. References

Testcase database, scripts and references can be found at 'Attachments' and 'Related Solutions' sections below the PDF on Cadence Support i.e. https://support.cadence.com, This pdf can be searched ONLINE with the document 'Title' as well.