ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and Electrical Engineering

# VLSI III: Test and Fabrication of VLSI Circuits

227-0148-00L

## Exercise 6

# Sweep Test

Dr. F. K. Gürkaynak
Prof. Dr. H. Kaeslin

SVN Rev.: 1907
Last Changed: 2017-03-29

# 1 Introduction

In this training we will have a look on timing setup and sweep tests on IC tester Advantest SoCV93000 which is located in ETZ J71.1. A preconfigured setting will be provided. Additional material can be found on our eda wiki, which is accessible from the ETH network on *ee.ethz.ch/index.php/Tester*.

# 2 Setup

First, lets start with the setup.

**Student Task 1:**

- We are using the ”Pony” chip, if it is not already in the tester room, you can get it in ETZ J60.1
- Set up the tester as usual or consider the eda-wiki article ’Tester: Basic Handling’. Pony uses a QFN56 package, thus mount tester board *Student_QFN56.std_v1_rev0*.
- Copy files to your exercise folder:

```
sh> cd YOUR_EXERCISE_FOLDER
sh> /home/vlsi3/ex06/install_ex06 .
```

- Start the SmarTest Eclipse Workcenter

```
sh> ~hp93000/bin/start_93000
```

- Select a temporary folder for the current session.
- Load the device. (`93000→Device→Change Device`).
  Select the according HP93000 folder from the *ex_06* folder.
- Load the preconfigured settings to the current Test Program (right-click and load on the according items in the test program explorer)
  - Pin configuration
  - Levels
  - Timing
  - Patterns

# 3 Timing measurements

The timing setup defines when and how the tester is proposed to apply the stimuli vectors and how the resulting output signals should be interpreted. The timing setup has three main components:

- **Wave Tables** which explain how to apply or interpret the signals.
- **Timing Equations** which define when to apply or interpret based on parameters and equations.

- **Timing Specifications** which are the actual parameter (e.g. frequency, supply voltage, ...); they can be modified in the timing spec window or in the testflow.

### 3.1 Wave Tables

**Student Task 2:** We will start with the wave tables:

- Open the timing setup window. For this mark the timing item in the Test Program Explorer and press `F3` (or double-click on it). The Timing Setup window appears.
- In the next step open the wave tables which can be found in the `Select`-menu.

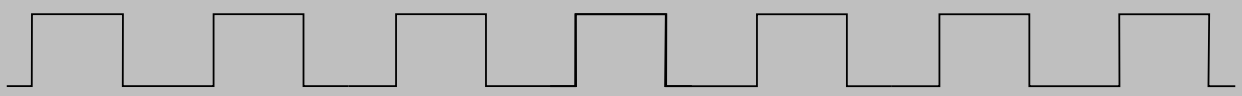
A wave table contains entries for different pin groups. In the first wave table a reasonable configuration for a standard ATI timing can be found. For an input pin the first column defines the wave table ID, the second defines what should be applied at different drive events (d1, d2, ...) and the third element indicates the according value in the stimuli file. The output pins define receive events (r1, r2, ...) and the values which are compared to the expected responses. Acquaint yourself with the Wave Table file.

**Important Note:** If you change the wave table, you need to save and download it to tester, because otherwise the change is just done locally. You can find it in the `Shell`-Menu.

### 3.2 Timing Equations

The timing equations define at which time the events defined in the wave tables should be executed. First lets recall the general ATI timing.

**Student Task 3:** Draw the drive/receive events in the following figure and discuss with the assistant.



**Student Task 4:**

- Open the Timing Equation editor which can be found in the Timing Setup window by selecting. `Select→edit equations`

In this file you can define different equation sets (EQNSET). Each equation is composed of:

- **SPECS:** Defines parameters which can be changed manually later or in the testflow and can be used in the equations.
- **EQUATIONS:** Defines values which are used for timing and are calculated from the specs.
- **TIMINGSETs:** Defines the timing of the drive and receive events for different pin groups (e.g. all output pins). Per equation set, multiple timing sets can be defined.

Have a look at the file and familiarise yourself with it.

- Complete the Simple/Functional timing set according to an ordinary ATI timing. Save it and download it to the tester.

### 3.3 Timing Specifications

**Student Task 5:** As last step the specification variables need to be set (which were defined in the timing equations). For this we will add three specification sets to the timing set "Simple". You can open the timing specs in the timing window by selecting `Select→edit specifications`. Create new spec sets with `Change→Create specification`. Give them reasonable names as this will help you in the following steps:

1. Specification 1: simple_150mhz with $f = 150\ \mathrm{MHz}$
2. Specification 2: simple_10mhz with $f = 10\ \mathrm{MHz}$ (already exists)
3. Specification 3: simple_dummy with $f = 100\ \mathrm{MHz}$

Select Equation Set "Simple", ID and description name as above. Then adjust the frequencies accordingly.
Do not forget to download them to the tester.
**Note:** as Tetramax uses an own timing scheme creating the ATPG dataset, this is taken into account in a specific Tetramax timing set.

## 4 Testflow

The timing sets are now defined and we can start with the tests.

**Student Task 6:**

- Load the testflow *testflow_vlsi3*
- Mount one of the available pony chips and connect.
- Run the basic tests: Continuity, ATPG and chip functionality tests.

All tests should work without errors.

### 4.1 Speed test

**Student Task 7:** The pony consists of several different crypto accelerators. To simplify the task we will only consider one accelerator which is the AEGIS crypto accelerator.

- Create a new Functional Test (ac_tml/AcTest/FunctionalTest) at the end of the testflow (by right-clicking on the dot and selecting `Insert→Run Test`) with the following setup:
  - Timing Equation: Simple, 10 MHz, Functional
  - Voltage Levels: Default, 2.5/1.2V, Standard
  - Patterns: aegisAll

- Run the test, does it work?
- Run the test with 150 MHz.
- If it failed, have a look at the Timing diagram.
  - You can find it in `93000→Results→Timing diagram`
  - Set the Display to scope size.
  - Go to the first errors by pressing several times *Next* until your reach the errors around the 9226th cycle and have a look at the exact output curve of *TREADY* by pressing *Scope*. Why does this chip not work under these conditions. Discuss with the assistant:

    _____

    _____

  - What could we do to reach the 150 MHz?

    _____

    _____

  - Have a look at the Standard and Ideal-Terminated level set and re-do the test with the adjusted levels. (By changing the level in the properties window of the created test).
  - Now change to the dummy specification and try if it works under this conditions with 165 and 175 MHz. You can change the specs in the specification window.

# 5 Sweep Tests

## 5.1 Speed Test

We could now try to find the maximum frequency by testing all frequencies by hand, but as you can imagine, it is not convenient to do this for each chip and we want to do this in a automatic manner. For this purpose Sweep Tests (also called SpecSearch) are used.

The idea is to change one parameter while the others are kept constant. To find the maximum/minium value there are two methods which can be used: linear or binary search. In the linear mode we define a minimal $f_{min}$, a maximal value $f_{max}$ and a step size $\Delta$. The tester will run the test with the frequencies $f_{min} + i \cdot \Delta < f_{max}$ or until it fails.

Alternatively the binary mode uses binary search where the search interval (initial: $f_{max} - f_{min}$) is split into two time intervals and the mean value is used for the test. If the test has passed the tester changes the search interval to the second half or the first half if it has failed. This new search interval is used to redo the first step iteratevely.

**Student Task 8:** Which requirements are needed to be sure to find the maximum frequency?

- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____

**Student Task 9:** You now know the basics to create a sweep test for the maximum frequency.

- Create a new test in the testflow. *Right-click* on the end of the testflow, then select `Insert→Run Test`.
- Select a suitable test suite name
- Choose the test method `ac_tml→AcTest→SpecSearch`
- Select a proper timing and level setup
- Select the patterns including the whole dataset for AEGIS.
- Select your new test.
- Adjust the spec you are looking for.
- Select the search method.
- Define the min and max values (and if you choose linear mode as well the step size)
- Select the right pins in the pin list
- Set Output to ReportUI
- Run the test.

What is the actual maximum frequency?

$f_{max} =$ _____

## 5.2 IO Timing

In this part of the exercise you will setup tests for IO Timing.

- Setup Time
- Hold Time
- Contamination Delay
- Propagation Delay

**Student Task 10:**

- Lets start with a safe setup. Draw in Figure 1 the timing events for input pins and output pins for cycle *k*.
- How can we now find the setup time? In Figure 1 draw the application and test events. Draw the event and indicate the range in which the parameter is sweeped.
- Although we have test patterns including the most critical setup path, we might not see this setup violation in the sweep test. Why?

  _____

  _____

- What could be a reasonable solution for this problem?

  _____

- Now draw in Figure 1 the input signal for the pattern sequence 01100 and the according test time.
- How could we find the contamination delay. Draw them in the timing plot.

**Student Task 11:** In this last practical task you will create the wave tables, timing equations, timing specs and the sweep tests for the io delays.

- Create the wave tables with the scheme discussed in the previous task. Define the wave tables according to the pin groups. (Hint: There are two pin groups comp_in and no_comp_in where the latter one consists of test pins and reset)
- Set up save timing specs.
- Create the tests and fill in the timing delays:

  $t_{setup} =$ _____ $t_{hold} =$ _____

  $t_{prop} =$ _____ $t_{cont} =$ _____

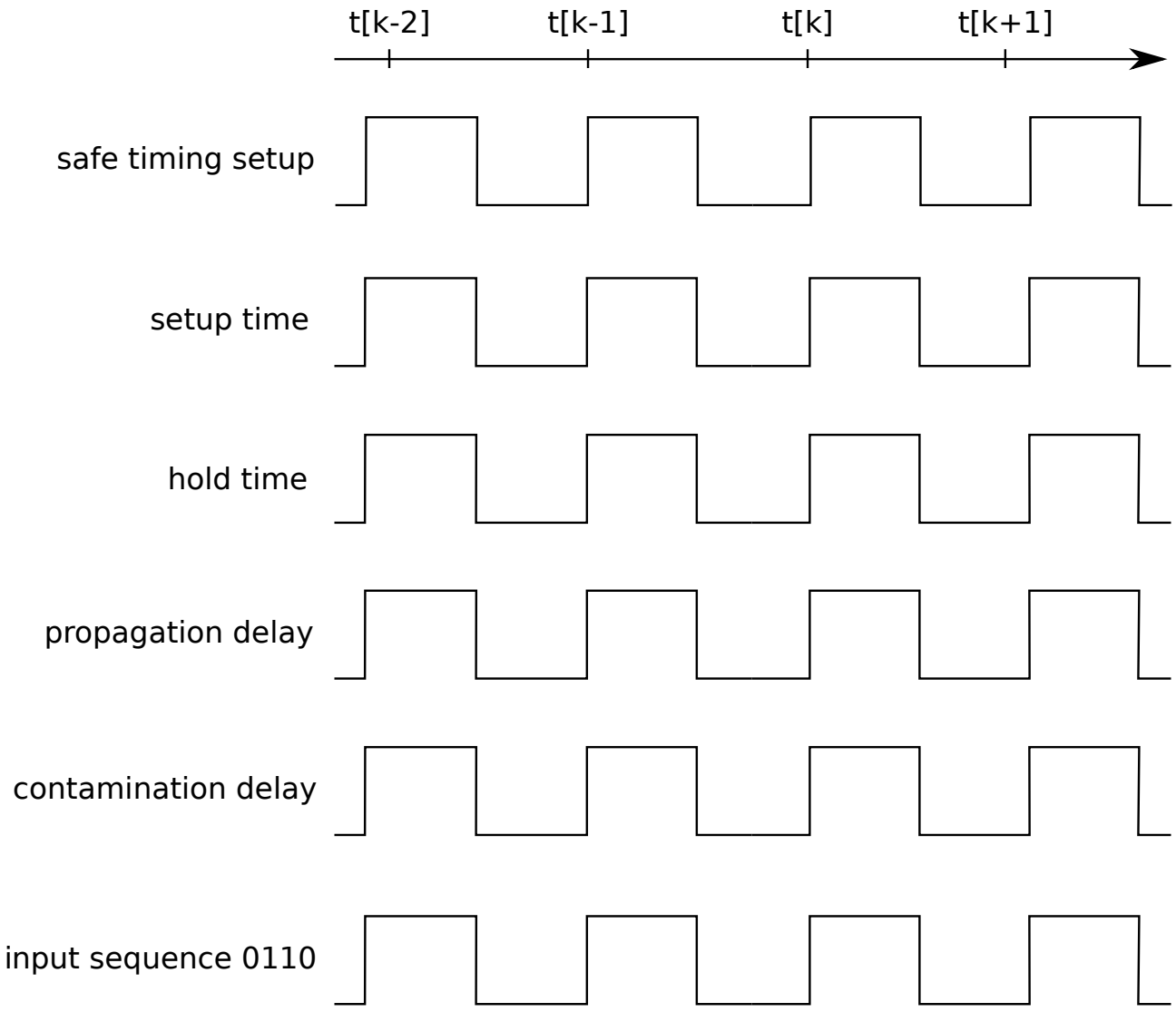



Figure 1: Timing events for IO timing tests.